NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

REAL-TIME SPEAKER DETECTION FOR USER-DEVICE BINDING

by

Mark J. Bergem

December 2010

Thesis Advisor:	Dennis Volpano
Second Reader:	Robert Beverly

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
21–12–2010	Master's Thesis	2008-12-01—2010-12-07

4. TITLE AND SUBTITLE

Real-Time Speaker Detection for User-Device Binding

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Mark J. Bergem

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. **IRB Protocol Number: XXXX**

14. ABSTRACT

This thesis explores the accuracy and utility of a framework for recognizing a speaker by his or her voice called the Modular Audio Recognition Framework (MARF). Accuracy was tested with respect to the MIT Mobile Speaker corpus along three axes: 1) number of training sets per speaker, 2) testing sample length and 3) environmental noise. Testing showed that the number of training samples per speaker had little impact on performance. It was also shown that MARF was successful using testing samples as short as 1000ms. Finally, testing discovered that MARF had difficulty with testing samples containing significant environmental noise.

An application of MARF, namely a referentially-transparent calling service, is described. Use of this service is considered for both military and civilian applications, specifically for use by a Marine platoon or a disaster-response team. Limitations of the service and how it might benefit from advances in hardware are outlined.

15. SUBJECT TERMS

Speaker Recognition,Voice,Biometrics,Referential Transparency,Cellular phones,mobile communication, military communications, disaster response communications

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	75	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

REAL-TIME SPEAKER DETECTION FOR USER-DEVICE BINDING

Mark J. Bergem
Lieutenant, Junior Grade, United States Navy
B.A. UC Santa Barbara

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2010

Author: Mark J. Bergem

Approved by: Dennis Volpano
Thesis Advisor

Robert Beverly
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis explores the accuracy and utility of a framework for recognizing a speaker by his or her voice called the Modular Audio Recognition Framework (MARF). Accuracy was tested with respect to the MIT Mobile Speaker corpus along three axes: 1) number of training sets per speaker, 2) testing sample length and 3) environmental noise. Testing showed that the number of training samples per speaker had little impact on performance. It was also shown that MARF was successful using testing samples as short as 1000ms. Finally, testing discovered that MARF had difficulty with testing samples containing significant environmental noise.

An application of MARF, namely a referentially-transparent calling service, is described. Use of this service is considered for both military and civilian applications, specifically for use by a Marine platoon or a disaster-response team. Limitations of the service and how it might benefit from advances in hardware are outlined.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

The roll-out of commercial wireless networks continues to rise worldwide. Growth is especially vigorous in under-developed countries as wireless communication has been a relatively cheap alternative to wired infrastructure.[2] With their low cost and quick deployment, it makes sense to explore the viability of stationary and mobile cellular networks to support applications beyond the current commercial ones. These applications include tactical military missions as well as disaster relief and other emergency services. Such missions often are characterized by relatively-small cellular deployments (on the order of fewer than 100 cell users) compared to commercial ones. How well suited are commercial cellular technologies and their applications for these types of missions?

Most smart-phones are equipped with a Global Positioning System (GPS) receiver. We would like to exploit this capability to locate individuals. But GPS alone is not a reliable indicator of a person's location. Suppose Sally is a relief worker in charge of an aid station. Her smart-phone has a GPS receiver. The receiver provides a geo-coordinate to an application on the device that in turn transmits it to you, perhaps indirectly through some central repository. The information you receive is the location of Sally's phone, not the location of Sally. Sally may be miles away if the phone was stolen, or worse, in danger and separated from her phone. Relying on GPS alone may be fine for targeted advertising in the commercial world, but it is unacceptable for locating relief workers without some way of physically binding them to their devices.

Suppose a Marine platoon (roughly 40 soldiers) is issued smartphones to communicate and learn the location of each other. The platoon leader receives updates and acknowledgments to orders. Squad leaders use the devices to coordinate calls for fire. During combat, a smartphone may become inoperable. It may be necessary to use another member's smartphone. Smartphones may also get switched among users by accident. So the geo-coordinates reported by these phones may no longer accurately convey the locations of the Marines to whom they were originally issued. Further, the platoon leader will be unable to reach individuals by name unless there is some mechanism for updating the identities currently tied to a device.

The preceding examples suggest at least two ways commercial cellular technology might be improved to support critical missions. The first is dynamic, physical *binding* of one or more

users to a cellphone. That way if we have the phone's location, we have the location of its users as well.

The second way is *calling by name*. We want to call a user, not a cellphone. If there is a way to *dynamically* bind a user to whatever cellphone they are currently using then we can always reach that user through a mapping of their name to a cell number. This is the function of a Personal Name System (PNS) analogous to the Domain Name System. Personal name systems are not new. They have been developed for general personal communications systems such as the Personal Communication System[3] developed at Stanford in 1998 [4]. Also, a PNS system is available as an add on for Avaya's Business Communications Manager PBX. A PNS is particularly well suited for small missions since these missions tend to have relatively small name spaces and fewer collisions among names. A PNS setup within the scope of this thesis is discussed in Chapter 4.

Another advantage of a PNS is that we are not limited to calling a person by their name but instead can use an alias. For example, alias *AidStationBravo* can map to *Sally*. Now should something happen to Sally, the alias could be quickly updated with her replacement without having to remember the change in leadership at that station. Moreover with such a system, broadcast groups can easily be implemented. We might have *AidStationBravo* maps to *Sally* and *Sue*, or even nest aliases as in *AllAidStations* maps to *AidStationBravo* and *AidStationAlpha*. Such aliasing is also very beneficial in the military setting where an individual can be contacted by a pseudonym rather than a device number. All members of a squad can be reached by the squad's name and so on.

The key to the improvements mentioned above is technology that allows us to *passively* and *dynamically* bind an identity to a cellphone. Biometrics serves this purpose.

1.1 Biometrics

Humans rely on biometrics to authenticate each other. Whether we meet in person or converse by phone, our brain distills the different elements of biology available to us (hair color, eye color, facial structure, vocal cord width and resonance, etc.) in order to authenticate a person's identity. Capturing, or "reading," biometric data is the process of capturing information about a biological attribute of a person. This attribute is used to create measurable data that can be used to derive unique properties of a person that is stable and repeatable over time and over variations in acquisition conditions. [5]

Use of biometrics has key advantages:

- Biometric is always with the user, there is no hardware to lose.
- Authentication may be accomplished with little or no input from the user.
- There is no password or sequence for the operator to forget or misuse.

What type of biometric is appropriate for binding a user to a cell phone? It would seem that a fingerprint reader might be ideal. After all, we are talking on a *hand-held* device. However, users often wear gloves, latex or otherwise. It would be an inconvenience to remove one's gloves every time they needed to authenticate to the device. Dirt, dust, and sweat can foul up a fingerprint scanner. Further, the scanner most likely would have to be an additional piece of hardware installed on the mobile device.

Fortunately, there are other types of biometrics available to authenticate users. Iris scanning is the most promising since the iris "is a protected internal organ of the eye, behind the cornea and the aqueous humour, it is immune to the environment except for its pupillary reflex to light. The deformations of the iris that occur with pupillary dilation are reversible by a well defined mathematical transform[6]". Accurate readings of the iris can be taken from one meter away. This would be a perfect biometric for people working in many different environments under diverse lighting conditions; from pitch black to searing sun. With a quick "snap-shot" of the eye we can identify our user. But how would this be installed in the device? Many smart-phones have cameras, but are they high enough quality to sample the eye? Even if the cameras are adequate, one still has to stop what they are doing to look into a camera. This is not as passive as we would like.

Work has been done on the use of body chemistry as a type of biometric. This can take into account things like body odor and body pH levels. This technology is promising as it could allow passive monitoring of the user while the device is worn. The drawback is this technology is still in the experimentation stage. There has been, to date, no actual system built to "smell" human body odor. The monitoring of pH is farther along and already in use in some medical devices, but these technologies still have yet to be used in the field of user identification. Even if the technology existed, how could it be deployed on a mobile device? It is reasonable to assume that a smart-phone will have a camera, it is quite another thing to assume it will have

an artificial "nose." Use of these technologies would only compound the problem. While they would be passive, they would add another piece of hardware into the chain.

None of the biometrics discussed so far meets our needs. They either can be foiled too easily, require additional hardware or are not as passive as they should be. There is an alternative that seems promising: speech. Speech is a passive biometric that naturally fits a cellphone. It does not require any additional hardware. One should not confuse speech with speech recognition which has had limited success in situations where there is significant ambient noise. Speech recognition is an attempt to understand what was spoken. Speech is merely sound that we wish to analyze and attribute to a speaker. This is called *speaker recognition*.

1.2 Speaker Recognition

Speaker recognition is the problem of analyzing a *testing sample* of audio and attributing it to a speaker. The attribution requires that a set of *training samples* be gathered before submitting testing samples for analysis. It is the training samples against which the analysis is done. A variant of this problem is called *open-set* speaker recognition. In this problem, analysis is done on a testing sample from a speaker for whom there are no training samples. In this case, the analysis should conclude the testing sample comes from an unknown speaker. This tends to be harder than closed-set recognition.

There are some limitations to overcome before speaker recognition becomes a viable way to bind users to cellphones. First, current implementations of speaker recognition degrade substantially as we increase the number of users for whom training samples have been taken. This increase in samples increases the confusion in discriminating among the registered speaker voices. In addition, this growth also increases the difficulty in confidently declaring a test utterance as belonging to or not belonging to the initially nominated registered speaker[7].

Question. Is population size a problem for our missions? For relatively small training sets, on the order of 40-50 people, is the accuracy of speaker recognition acceptable?

Speaker recognition is also susceptible to environmental variables. Using the latest feature extraction technique (MFCC explained in the next chapter) one sees nearly a 0% failure rate in quiet environments in which both training and testing sets are gathered [8]. Yet the technique is highly vulnerable to noise, both ambient and digital.

Question. How does the technique perform under our conditions?

Speaker recognition requires a training set to be pre-recorded. If both the training set and testing sample are made in a similar noise-free environment, speaker recognition can be quite successful.

Question. What happens when testing and training samples are taken from environments with different types and levels of ambient noise?

This thesis aims to answer the preceding questions using an open-source implementation of MFCC called Modular Audio Recognition Framework (MARF). We will determine how well the MARF platform performs in the lab. We will look not only at the baseline "clean" environment, where both the recorded voices and testing samples are made in noiseless environments, but we shall examine the injection of noise into our samples. The noise will come both from the ambient background of the physical environment and the digital noise created by packet loss, mobile device voice codecs, and audio compression mechanisms. We shall also examine the shortcomings with MARF and how, due to platform limitations, we were unable improve upon our results.

1.3 Thesis Roadmap

We will begin with some background, specifically some history behind, and methodologies for, speaker recognition. Next, we will explore both the evolution and state of the art of speaker recognition. Then we will look at what products currently support speaker recognition and why we decided on MARF for our recognition platform.

Next we will investigate an architecture in which to host speaker recognition. We will look at the trade-offs of deploying on a mobile device versus on a server. Which is more robust? How scalable is it? We propose one architecture for the system and explore uses for it. Its military applications are apparent, but its civilian applications could have significant impact on the efficiency of emergency response teams and the ability to quickly detect and locate missing personnel. From Army companies to small tactical team, from regional disaster response to six-man SWAT teams, this system can be quickly re-scaled to meet very diverse needs.

Lastly, we will look at where we go from here. What are the major shortcomings with our approach? We will examine which issues can be solved with the application of this new software and which ones need to wait for advances in hardware. We will explore which areas of research need to be further developed to bring advances in speaker recognition. Finally, we examine "spin-offs" of this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Speaker Recognition

2.1 Speaker Recognition

2.1.1 Introduction

As we listen to people, we are innately aware that no two people sound alike. This means aside from the information that the person is actually conveying through speech, there is other data, metadata if you will, that is sent along that tells us something about *how* they speak. There is some mechanism in our brain that allows us to distinguish between different voices, much as we do with faces or body appearance. Speaker recognition in software is the ability to make machines do what is automatic for us. The field of speaker recognition has been around for quite sometime, but with the explosion of computation power within the last decade we have seen significant growth in the field.

The speaker recognition problem has two inputs: a voice sample, also called a *testing sample*, and a set of *training samples* taken from a training group of speakers. If the testing sample is known to have come from one of the speakers in the training group then identifying *which* one is called *closed-set* speaker recognition. If the testing sample may be drawn from a speaker population outside the training group then recognizing when this is so, or identifying which speaker uttered the testing sample when it is not, is called *open-set* speaker recognition [9]. A related but different problem is *speaker verification*, also know as speaker authentication or detection. In this case, the problem is given a testing sample and alleged identity as inputs, verifying the sample originated from the speaker with that identity. In this case, we assume that any impostors to the system are not known to the system so the problem is open-set recognition.

Important to the speaker recognition problem are the training samples. One must decide whether the phrases to be uttered are *text-dependent* or *text-independent*. With a system that is text-dependent, the same phrase is uttered by a speaker in both the testing and training samples. While text-dependent recognition yields higher success rates [10], voice samples for our purposes are text independent. Though less accurate, text independence affords biometric passivity and allows us to use shorter sample sizes since we do not need to sample an entire word or passphrase.

Below are the high-level steps of an algorithm for open-set speaker recognition [11]:

1. enrollment, or first recording of our users, generating speaker reference models
2. digital speech data acquisition
3. feature extraction
4. pattern matching
5. accepting or rejecting

Joseph Campbell lays this process out well in his paper:

> Feature extraction maps each interval of speech to a multidimensional feature space. (A speech interval typically spans 1030 ms of the speech waveform and is referred to as a frame of speech.) This sequence of feature vectors x_i is then compared to speaker models by pattern matching. This results in a match score for each vector or sequence of vectors. The match score measures the similarity of the computed input feature vectors to models of the claimed speaker or feature vector patterns for the claimed speaker. Last, a decision is made to either accept or reject the claimant according to the match score or sequence of match scores, which is a hypothesis-testing problem.[11]

Looking at the work done by MIT with the corpus used in Chapter 3 we can get an idea of what results we should expect. MIT's testing varied slightly as they used Hidden Markov Models (HMM) (explained below) which is not supported by MARF.

They initially tested with mismatched conditions. In particular, they examined the impact of environment and microphone variability inherent with handheld devices [12]. Their results are as follows:

> System performance varies widely as the environment or microphone is changed between the training and testing phase. While the fully matched trial (trained and tested in the office with an earpiece headset) produced an equal error rate (EER) of 9.4%, moving to a matched microphone/mismatched environment (trained in

a lobby with the earpiece microphone but tested at a street intersection with an earpiece microphone) resulted in a relative degradation in EER of over 300% (EER of 29.2%). [12]

In Chapter 3 we will put these results to the test and see how MARF, using different feature extraction and pattern matching than MIT, fares with mismatched conditions.

2.1.2 Feature Extraction

What are these features of voice that we must unlock to have the machine recognize the person speaking? Though there are no set features that we can examine, source-filter theory tells us that the sound of speech from the user must encode information about their own vocal biology and pattern of speech. Therefore, using short-term signal analysis, say in the realm of 10ms-20ms, we can extract features unique to a speaker. This is typically done with either FFT analysis or LPC (all-pole) to generate magnitude spectra which are then converted to melcepstrum coefficients [10]. If we let x be a vector that contains N sound samples; mel-cepstrum coefficients are obtained by the following computation:[13]

- Discrete Fourier transform (DFT) $\hat{x}$ of the data vector x is computed using the FFT algorithm and a Hanning window.
- The DFT $(\hat{x})$ is divided into M nonuniform subbands, and the energy $(e_i, i = 1, 2, ..., M)$ of each subband is estimated. The energy of each subband is defined as $e_i = \sum_{l=p}^{q}$, where p and q are the indices of subband edges in the DFT domain. The subbands are distributed across the frequency domain according to a "melscale," which is linear at low frequencies and logarithmic thereafter. This mimics the frequency resolution of the human ear. Below 10 kHz, the DFT is divided linearly into 12 bands. At higher frequency bands, covering 10 to 44 kHz, the subbands are divided in a logarithmic manner into 12 sections.
- The melcepstrum vector (c = [c1, c2,..., cK]) is computed from the discrete cosine transform (DCT):

  $c_k = \sum_{i=1}^{M} \log(e_i) \cos[k(i - 0.5)\pi/M], k = 1, 2, \cdots K$

  where the size of the melcepstrum vector (K) is much smaller than data size N. [13]

These vectors will typically have 24-40 elements.

Fast Fourier Transform (FFT)

The Fast Fourier Transform (FFT) algorithm is used both for feature extraction and as the basis for the filter algorithm used in preprocessing. Essentially the FFT is an optimized version of the Discrete Fourier Transform. It takes a window of size 2^k and returns a complex array of coefficients for the corresponding frequency curve. For feature extraction, only the magnitudes of the complex values are used, while the FFT filter operates directly on the complex results. The implementation involves two steps: First, shuffling the input positions by a binary reversion process, and then combining the results via a "butterfly" decimation in time to produce the final frequency coefficients. The first step corresponds to breaking down the time-domain sample of size n into n frequency- domain samples of size 1. The second step re-combines the n samples of size 1 into 1 n-sized frequency-domain sample.[1]

FFT Feature Extraction The frequency-domain view of a window of a time-domain sample gives us the frequency characteristics of that window. In feature identification, the frequency characteristics of a voice can be considered as a list of "features" for that voice. If we combine all windows of a vocal sample by taking the average between them, we can get the average frequency characteristics of the sample. Subsequently, if we average the frequency characteristics for samples from the same speaker, we are essentially finding the center of the cluster for the speaker's samples. Once all speakers have their cluster centers recorded in the training set, the speaker of an input sample should be identifiable by comparing her frequency analysis with each cluster center by some classification method. Since we are dealing with speech, greater accuracy should be attainable by comparing corresponding phonemes with each other. That is, "th" in "the" should bear greater similarity to "th" in "this" than will "the" and "this" when compared as a whole. The only characteristic of the FFT to worry about is the window used as input. Using a normal rectangular window can result in glitches in the frequency analysis because a sudden cutoff of a high frequency may distort the results. Therefore it is necessary to apply a Hamming window to the input sample, and to overlap the windows by half. Since the Hamming window adds up to a constant when overlapped, no distortion is introduced. When comparing phonemes, a window size of about 2 or 3 ms is appropriate, but when comparing whole words, a window size of about 20 ms is more likely to be useful. A larger window size produces a higher resolution in the frequency analysis.[1]

Linear Predictive Coding (LPC)

LPC evaluates windowed sections of input speech waveforms and determines a set of coefficients approximating the amplitude vs. frequency function. This approximation aims to repli-

cate the results of the Fast Fourier Transform yet only store a limited amount of information: that which is most valuable to the analysis of speech.[1]

The LPC method is based on the formation of a spectral shaping filter, $H(z)$, that, when applied to a input excitation source, $U(z)$, yields a speech sample similar to the initial signal. The excitation source, $U(z)$, is assumed to be a flat spectrum leaving all the useful information in $H(z)$. The model of shaping filter used in most LPC implementation is called an "all-pole" model, and is as follows:

$$H(z) = \frac{G}{(1-\sum_{k=1}^{p}(a_k z^{-k}))}$$

Where p is the number of poles used. A pole is a root of the denominator in the Laplace transform of the input-to-output representation of the speech signal.[1]

The coefficients a_k are the final representation of the speech waveform. To obtain these coefficients, the least-square autocorrelation method was used. This method requires the use of the auto-correlation of a signal defined as:

$$R(k) = \sum_{m=k}^{n-1}(x(n) \cdot x(n-k))$$

where x(n) is the windowed input signal.[1]

In the LPC analysis, the error in the approximation is used to derive the algorithm. The error at time n can be expressed in the following manner: $e(n) = s(n)\sum_{k=1}^{p}(a_k \cdot s(n-k))$. Thus, the complete squared error of the spectral shaping filter $H(z)$ is:

$$E = \sum_{n=-\infty}^{\infty}(x(n) - \sum_{k=1}^{p}(a_k \cdot x(nk)))$$

To minimize the error, the partial derivative $\frac{\partial E}{\partial a_k}$ is taken for each $k = 1..p$, which yields p linear equations in the form:

$$\sum_{n=-\infty}^{\infty}(x(n-1) \cdot x(n)) = \sum_{k=1}^{p}(a_k \cdot \sum_{n=-\infty}^{\infty}(x(n-1) \cdot x(n-k))$$

For $i = 1..p$. Which, using the auto-correlation function is:

$$\sum_{k=1}^{p}(a_k \cdot R(i-k)) = R(i)$$

Solving these as a set of linear equations and observing that the matrix of auto-correlation values is a Toeplitz matrix yields the following recursive algorithm for determining the LPC coefficients:

$$k_m = \frac{R(m) - \sum_{k=1}^{m-1}(a_{m-1}(k)R(m-k))}{e_{m-1}}$$
$$a_m(m) = k_m$$
$$a_m(k) = a_{m-1}(k) - k_m \cdot a_m(m-k) \text{ for } 1 \leq k \leq m-1$$
$$E_m = (1 - k_m^2) \cdot E_{m-1}$$

This is the algorithm implemented in the MARF LPC module.[1]

Usage in Feature Extraction The LPC coefficients are evaluated at each windowed iteration, yielding a vector of coefficients of the size p. These coefficients are averaged across the whole signal to give a mean coefficient vector representing the utterance. Thus a p sized vector was used for training and testing. The value of p chosen was based on tests given speed vs. accuracy. A p value of around 20 was observed to be accurate and computationally feasible.[1]

2.1.3 Pattern Matching

When the system trains a user, the voice sample is passed through the feature extraction process as discussed above. The vectors that are created are used to make the biometric *voice-print* of that user. Ideally we want the voice-print to have the following characteristics: "1) a theoretical underpinning so one can understand model behavior and mathematically approach extensions and improvements; (2) generalizable to new data so that the model does not over fit the enrollment data and can match new data; (3) parsimonious representation in both size and computation [9]."

The attributes of this training vector can be clustered to form a *code-book* for each trained user. So, when a new voice is sampled in the *testing* phase, the vector generated from the new voice sample is compared against the existing *code-books* of known users.

There are two primary ways to conduct pattern matching: stochastic models and template models. In stochastic models, the pattern matching is probabilistic and results in a measure of the

likelihood, or conditional probability, of the observation given the model. For template models, the pattern matching is deterministic [11].

The template model and its corresponding distance measure is perhaps the most intuitive since the template method can be dependent or independent of time. Common models used are: Chebyshev, or Manhattan Distance, Euclidean Distance, Minkowski Distance, and Mahalanobis Distance. Please see Section 2.2.3 for a detail description of how these algorithms are implemented in MARF.

The most common stochastic models used in speaker recognition are the Hidden Markov Models. They encode the temporal variations of the features and efficiently model statistical changes in the features, to provide a statistical representation of how a speaker produces sounds. During enrollment, HMM parameters are estimated from the speech using established algorithms. During verification, the likelihood of the test feature sequence is computed against the speaker's HMMs.[10] For text-independent applications, single state HMMs, also known as Gaussian Mixture Models (GMMs), are used. From published results, HMM based systems generally produce the best performance [9]. MARF does not support HMMs and therefore their experimentation is outside the scope of this thesis.

2.2 Modular Audio Recognition Framework

2.2.1 What is it?

MARF stands for Modular Audio Recognition Framework. It contains a collection of algorithms for Sound, Speech, and Natural Language Processing arranged into an uniform framework to facilitate addition of new algorithms for prepossessing, feature extraction, classification, parsing, etc. implemented in Java. MARF can give researchers a platform to test existing and new algorithms. The frameworks originally evolved around audio recognition, but research is not restricted to it due to MARF's generality as well as that of its algorithms [14].

MARF is not the only open source speaker recognition platform available. The author of this thesis examined both Alize [15] and CMU's Sphinx [16]. Sphinx, while promising for its support of HMM, is primary a speech recognition application. Its support for speaker recognition was almost non-existent. Alize, while a full featured speaker recognition toolkit, is written in the C programming language with the bulk of its user documentation written in French. This leaves MARF. A fully supported, well documented, language toolkit that supports speaker recognition. Also, MARF is written in Java, requiring no tweaking of the source code to run it on different

operating systems or hardware. Fulfilling the portable toolkit need as laid out in Chapter 5.

2.2.2 MARF Architecture

Before we begin, let us examine the basic MARF system architecture. Let us take a look at the general MARF structure in Figure 2.1.

The MARF class is the central "server" and configuration "placeholder", which contains the major methods for a typical pattern recognition process. The figure presents basic abstract modules of the architecture. When a developer needs to add or use a module, they derive from the generic ones.

A conceptual data-flow diagram of the pipeline is in Figure 2.2.

The gray areas indicate stub modules that are yet to be implemented. Consequently, the framework has the mentioned basic modules, as well as some additional entities to manage storage and serialization of the input/output data.

An application, using the framework, has to choose the concrete configuration and sub-modules for pre-processing, feature extraction, and classification stages. There is an API the application may use defined by each module or it can use them through the MARF.

2.2.3 Audio Stream Processing

While running MARF, the audio stream goes through three distinct processing stages. First there is the Pre-possessing filter. This modifies the raw wave file and prepares it for processing. After pre-processing, which may be skipped with the *raw* option, comes Feature Extraction. Here is where we see class feature extraction such as FFT and LPC. Finally, classification is run as the last stage.

Pre-processing

Pre-precessing is done to the sound file to prepare it for feature extraction. Ideally we want to normalize the sound or perform some type of filtering on it to remove excessive noise or interference. MARF supports most of the common audio pre-processing filters. These filter options are: `-raw`, `-norm`, `-silence`, `-noise`, `-endp`, and the following FFT filters: `-low`,`-high`, and `-band`. Interestingly, as shown in Chapter 3, the most successful filtering was no filtering at all, achieved in MARF by bypassing all preprocessing with the `-raw` flag. Figure 2.3, shows the API along with the description of the methods.

"Raw Meat", `-raw`

This is a basic "pass-everything-through" method that does not do actually do any pre-processing. Originally developed within the framework, it was meant to be a base line method, but it gives better top results out of many configurations including the testing done in Chapter 3. It it important to point out that this preprocessing method does not do any normalization. Further reseach should be done to show the effectivness, or detriment, of normalization. Likewise silence and noise removal is not done with this processing method.[1]

Normalization, `-norm`

Since not all voices will be recorded at exactly the same level, it is important to normalize the amplitude of each sample in order to ensure that features will be comparable. Audio normalization is analogous to image normalization. Since all samples are to be loaded as floating point values in the range $[-1.0, 1.0]$, it should be ensured that every sample actually does cover this entire range.[1]

The procedure is relatively simple: find the maximum amplitude in the sample, and then scale the sample by dividing each point by this maximum. Figure 2.4 illustrates normalized input wave signal.

Noise Removal, `-noise`

Any vocal sample taken in a less-than-perfect (which is always the case) environment will experience a certain amount of room noise. Since background noise exhibits a certain frequency characteristic, if the noise is loud enough it may inhibit good recognition of a voice when the voice is later tested in a different environment. Therefore, it is necessary to remove as much environmental interference as possible.[1]

To remove room noise, it is first necessary to get a sample of the room noise by itself. This sample, usually at least 30 seconds long, should provide the general frequency characteristics of the noise when subjected to FFT analysis. Using a technique similar to overlap-add FFT filtering, room noise can then be removed from the vocal sample by simply subtracting the frequency characteristics of noise from the vocal sample in question.[1]

Silence Removal, `-silence`

The silence removal is performed in time domain where the amplitudes below the threshold are discarded from the sample. This also makes the sample smaller and less similar to other samples thereby improving overall recognition performance.

The actual threshold can be set through a parameter, namely *ModuleParams*, which is a third parameter according to the pre-precessing parameter protocol.[1]

Endpointing, `-endp`

Endpointing the deciding where an utterenace begins and ends, then filtering out the rest of the stream as noise. The endpointing algorithm is implemented in MARF as follows. By the end-points we mean the local minimums and maximums in the amplitude changes. A variation of that is whether to consider the sample edges and continuous data points (of the same value) as end-points. In MARF, all these four cases are considered as end-points by default with an option to enable or disable the latter two cases via setters or the ModuleParams facility. [1]

FFT Filter

The Fast Fourier transform (FFT) filter is used to modify the frequency domain of the input sample in order to better measure the distinct frequencies we are interested in. Two filters are useful to speech analysis: high frequency boost and low-pass filter.[1]

Speech tends to fall off at a rate of 6 dB per octave, and therefore the high frequencies can be boosted to introduce more precision in their analysis. Speech, after all, is still characteristic of the speaker at high frequencies, even though they have a lower amplitude. Ideally this boost should be performed via compression, which automatically boosts the quieter sounds while maintaining the amplitude of the louder sounds. However, we have simply done this using a positive value for the filter's frequency response. The low-pass filter is used as a simplified noise reducer, simply cutting off all frequencies above a certain point. The human voice does not generate sounds all the way up to 4000 Hz, which is the maximum frequency of our test samples, and therefore since this range will only be filled with noise, it is common to just eliminate it. [1]

Essentially the FFT filter is an implementation of the Overlap-Add method of FIR filter design [17]. The process is a simple way to perform fast convolution, by converting the input to the frequency domain, manipulating the frequencies according to the desired frequency response, and then using an Inverse- FFT to convert back to the time domain. Figure 2.5 demonstrates the normalized incoming wave form translated into the frequency domain.[1]

The code applies the square root of the hamming window to the input windows (which are overlapped by half-windows), applies the FFT, multiplies the results by the desired frequency response, applies the Inverse-FFT, and applies the square root of the hamming window again,

to produce an undistorted output.[1]

Another similar filter could be used for noise reduction, subtracting the noise characteristics from the frequency response instead of multiplying, thereby removing the room noise from the input sample.[1]

Low-Pass, High-Pass, and Band-Pass Filters, `-low`,`-high`,`-band`

The low-pass filter has been realized on top of the FFT Filter, by setting up frequency response to zero for frequencies past a certain threshold chosen heuristically based on the window cut-off size. All frequencies past 2853 Hz were filtered out. See Figure 2.6.

As with the low-pass filter, the high-pass filter has been realized on top of the FFT Filter, in fact, it is the opposite to low-pass filter, and filters out frequencies before 2853 Hz. See Figure 2.7.

Finally, the band-pass filter in MARF is yet another instance of an FFT Filter, with the default settings of the band of frequencies of [1000, 2853] Hz. See Figure 2.8.[1]

Feature Extraction

Present here are the feature extraction algorithms used by MARF. Since both FFTs and LPCs are described above in Section 2.1.2, their detailed description will be left out from below. MARF fully supports both FFT and LPC feature extraction (`-fft`, `-lpc`). MARF also support feature extraction of Min/Max and a Feature Extraction Aggregation.

Hamming Window

Before we proceed with the other forms of feature extraction, let us briefly discuss "windowing". To extract the features from our speech, it it necessary to cut it up into smaller pieces as opposed to processing the whole sound file all at once. The technique of cutting a sample into smaller pieces to be considered individually is called "windowing". The simplest kind of window to use is the "rectangle", which is simply an unmodified cut from the larger sample.[1]

Unfortunately, rectangular windows can introduce errors, because near the edges of the window there will potentially be a sudden drop from a high amplitude to nothing, which can produce false "pops" and clicks in the analysis.[1]

A better way to window the sample is to slowly fade out toward the edges, by multiplying the points in the window by a "window function". If we take successive windows side by side, with the edges faded out, we will distort our analysis because the sample has been modified by

the window function. To avoid this, it is necessary to overlap the windows so that all points in the sample will be considered equally. Ideally, to avoid all distortion, the overlapped window functions should add up to a constant. This is exactly what the Hamming window does. It is defined as:

$$x(n) = 0.54 - 0.46 \cdot cos(\tfrac{2\pi n}{l-1})$$

where x is the new sample amplitude, n is the index into the window, and l is the total length of the window.[1]

Min/Max Amplitudes, `-minmax`

The Min/Max Amplitudes extraction simply involves picking up X maximums and N minimums out of the sample as features. If the length of the sample is less than X + N , the difference is filled in with the middle element of the sample.

This feature extraction does not perform very well yet in any configuration because of the simplistic implementation: the sample amplitudes are sorted and N minimums and X maximums are picked up from both ends of the array. As the samples are usually large, the values in each group are really close if not identical making it hard for any of the classifiers to properly discriminate the subjects. An improvement to MARF would be to pick up values in N and X distinct enough to be features and for the samples smaller than the $X + N$ sum, use increments of the difference of smallest maximum and largest minimum divided among missing elements in the middle instead one the same value filling that space in.[1]

Feature Extraction Aggregation, `-aggr`

This option by itself does not do any feature extraction, but instead allows concatenation of the results of several actual feature extractors to be combined in a single result. Currently in MARF, FFT and LPC are the extractors aggregated. Unfortunately, the main limitation of the aggregator is that all the aggregated feature extractors act with their default settings [1], that is to say, we cannot customize how we want each feature extrator to run when we invoke `-aggr`. Yet, interestingly, this method of feature extraction produces the best results with MARF.

Random Feature Extraction, `-randfe`

Given a window of size 256 samples, `-randfe` picks at random a number from a Gaussian distribution. This number is multiplied by the incoming sample frequencies. These numbers are combined to create a feature vector. This extraction is really based on no mechanics of

the speech, but really a random vector based on the sample. This should be the bottom line performance of all feature extraction methods. It can also be used as a relatively fast testing module.[1] Not surprisingly, this method of feature extraction produced extremely poor results.

Classification

Classification is the last step in the speaker verification process. After feature extraction, we a have mathematical representation of voice that can be mathematically compared to another vector. Since feature extraction is run on both our *learned* and *testing* samples, we have two vectors to compare. Classification gives us methods to perform this comparison.

Chebyshev Distance, `-cheb`

Chebyshev distance is used along with other distance classifiers for comparison. Chebyshev distance is also known as a city-block or Manhattan distance. Here is its mathematical representation:

$$d(x, y) = \sum_{k=1}^{n}(|x_k - y_k|)$$

where x and y are features vectors of the same length n.[1]

Euclidean Distance, `-eucl`

The Euclidean Distance classifier uses an Euclidean distance equation to find the distance between two feature vectors.

If $A = (x_1, x_2)$ and $B = (y_1, y_2)$ are two 2-dimensional vectors, then the distance between A and B can be defined as the square root of the sum of the squares of their differences:

$$d(x, y) = \sqrt{(x_2 - y_2)^2 + (x_1 - y_1)^2}$$

Minkowski Distance, `-mink`

Minkowski distance measurement is a generalization of both Euclidean and Chebyshev distances.

$$d(x, y) = (\sum_{k=1}^{n}(|x_k - y_k|)^r)^{\frac{1}{r}}$$

where r is a Minkowski factor. When $r = 1$, it becomes Chebyshev distance, and when $r = 2$, it is the Euclidean one. x and y are feature vectors of the same length n.[1]

Mahalanobis Distance, `-mah`

The Mahalanobis distance is based on weighting features with the inverse of their variance. Features with low variance are boosted and have a better chance of influencing the total distance. The Mahalanobis distance also involves an estimation of the feature covariances. Mahalanobis, given enough speech data, can generate more reliable variances for each vowel context, which can improve its performance [18].

$$d(x, y) = \sqrt{(x - y)C^{-1}(x - y)^T}$$

where x and y are feature vectors of the same length n, and C is a covariance matrix, learned during training for co-related features.[1] Mahalanobis distance was found to be a useful classifier in testing.

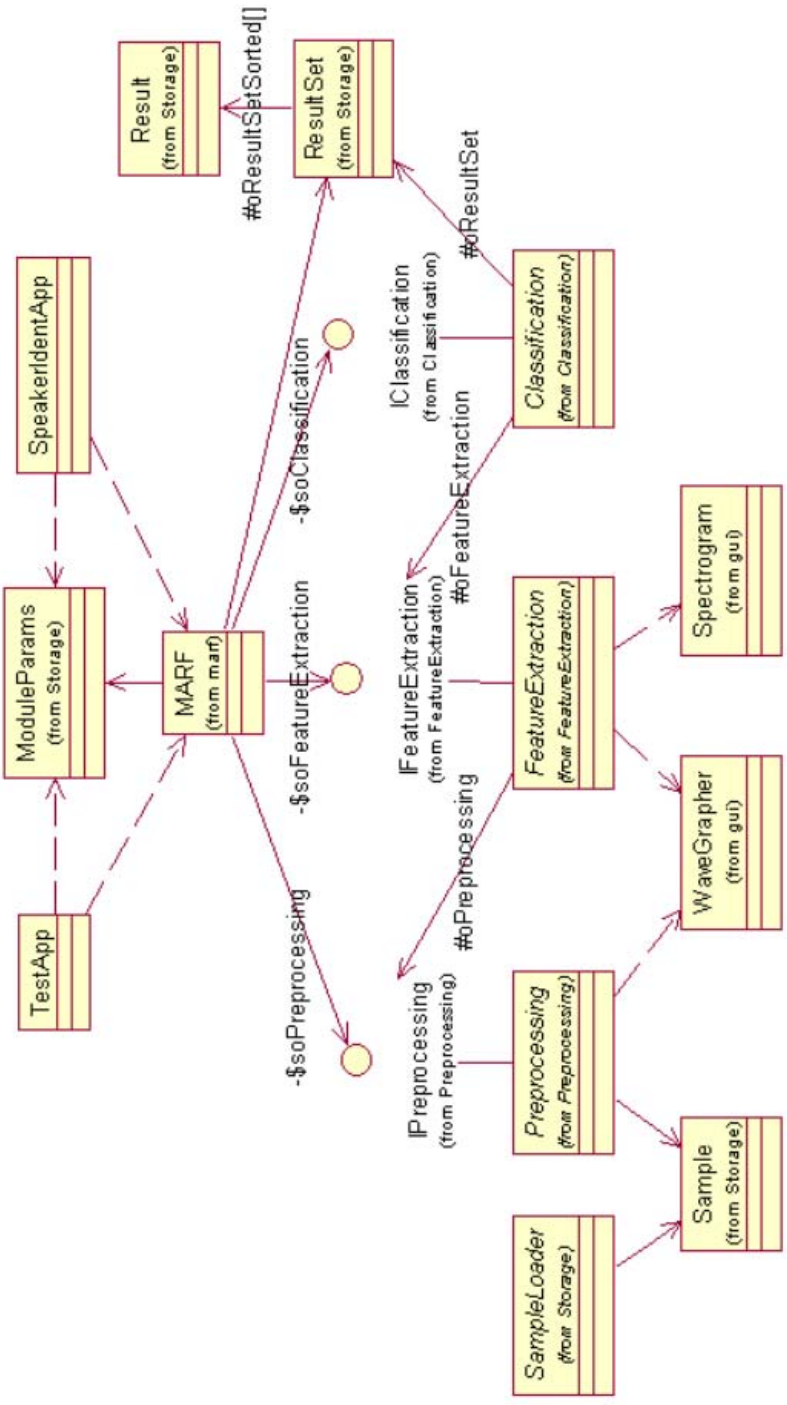

Figure 2.1: Overall Architecture [1]

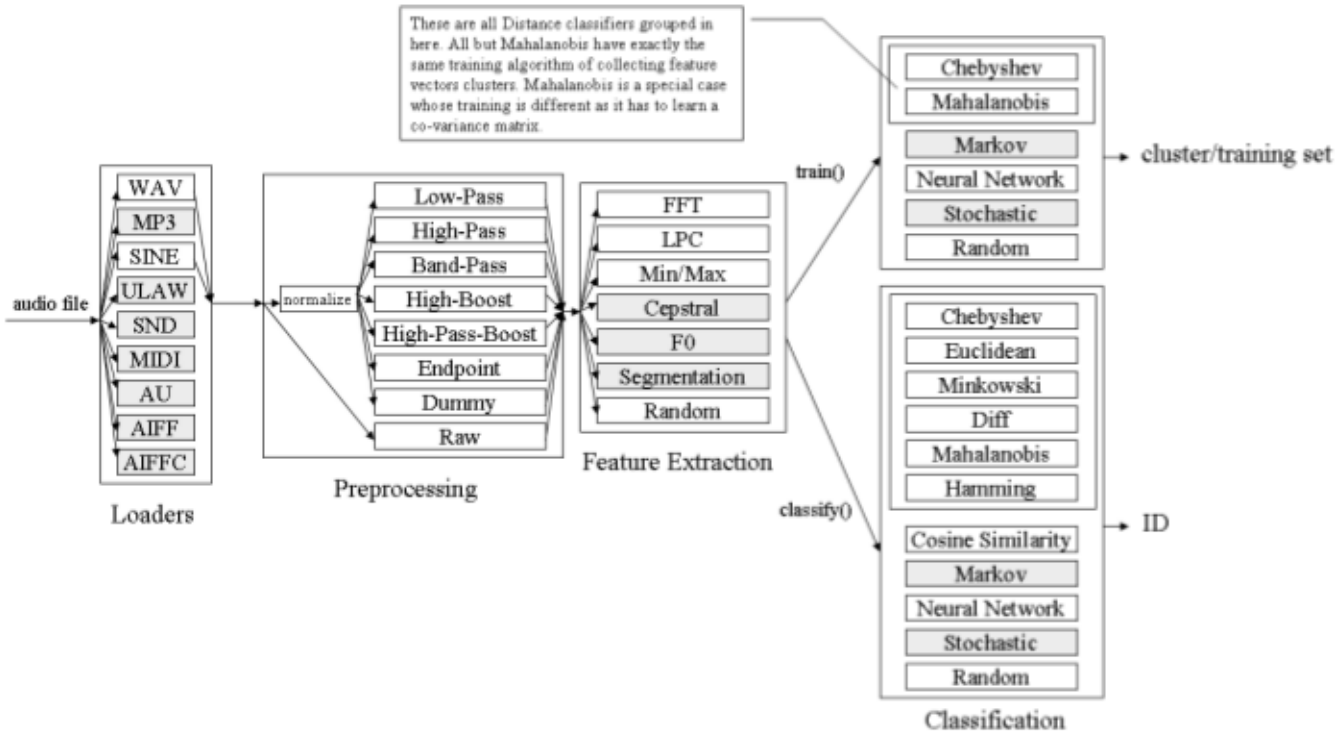

Figure 2.2: Pipeline Data Flow [1]

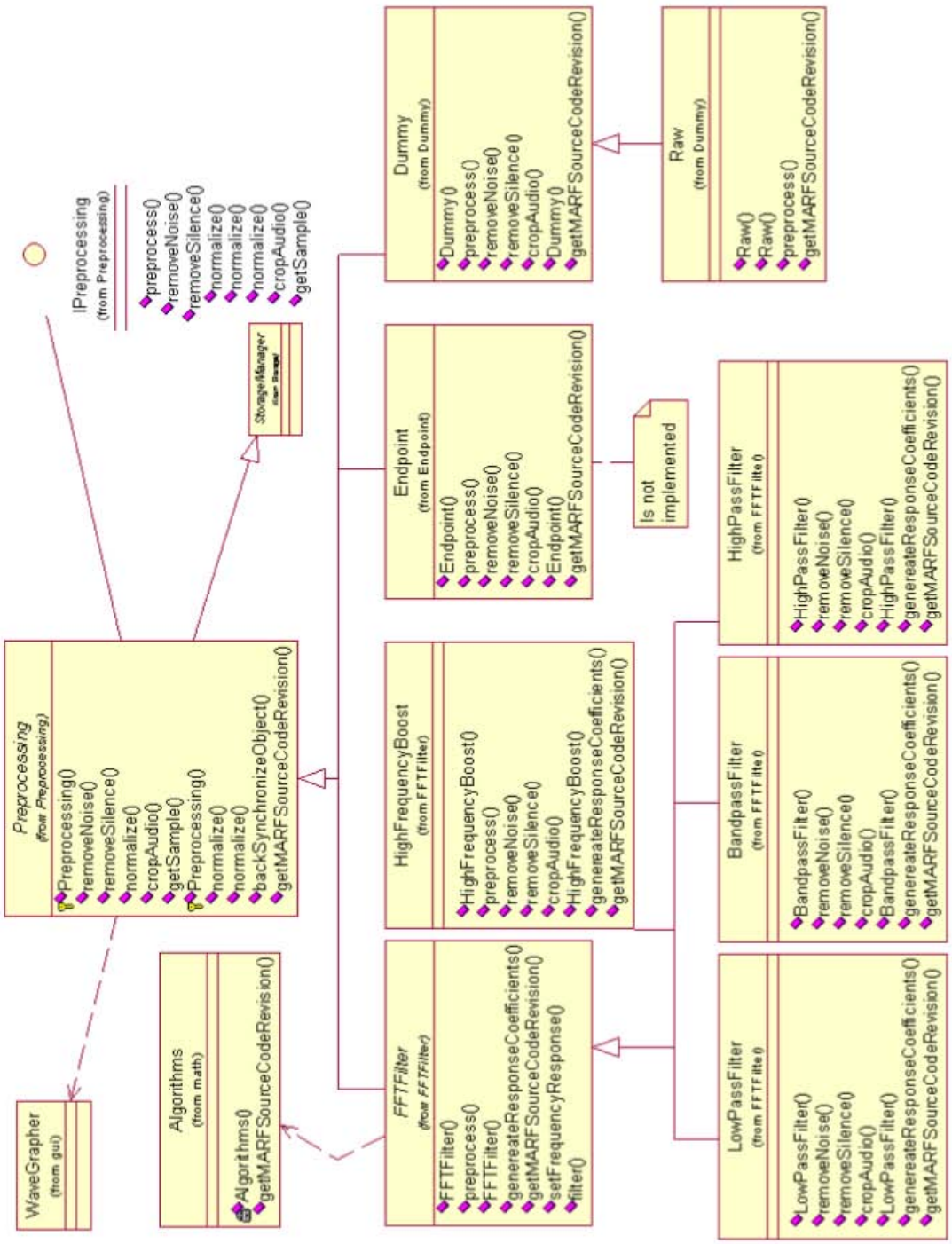

Figure 2.3: Pre-processing API and Structure [1]

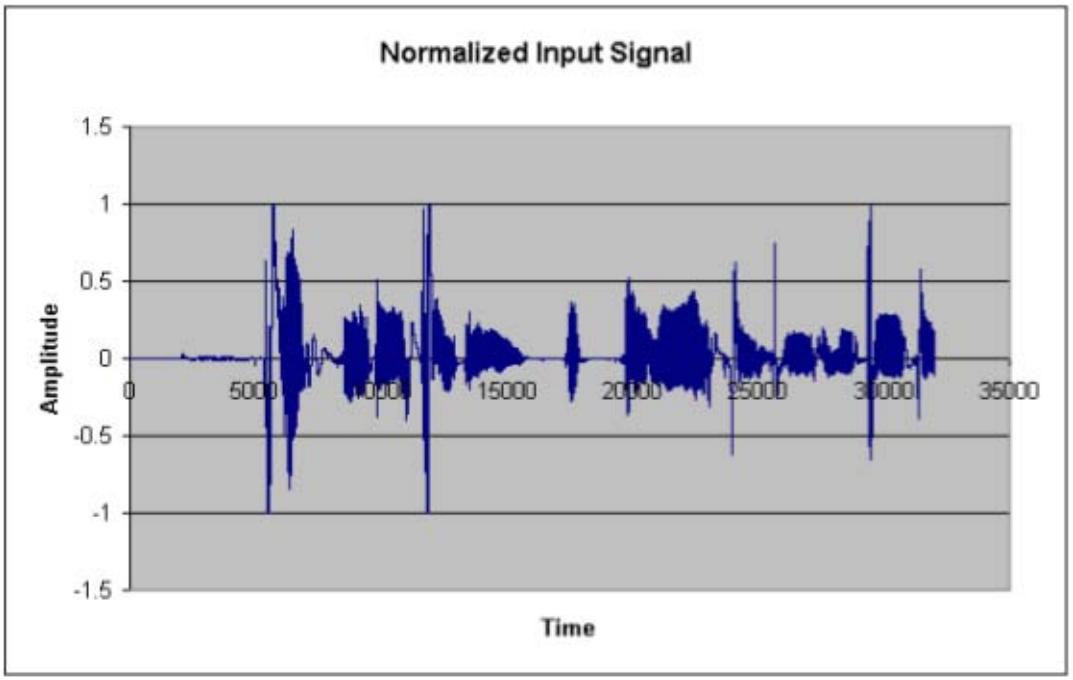

Figure 2.4: Normalization [1]

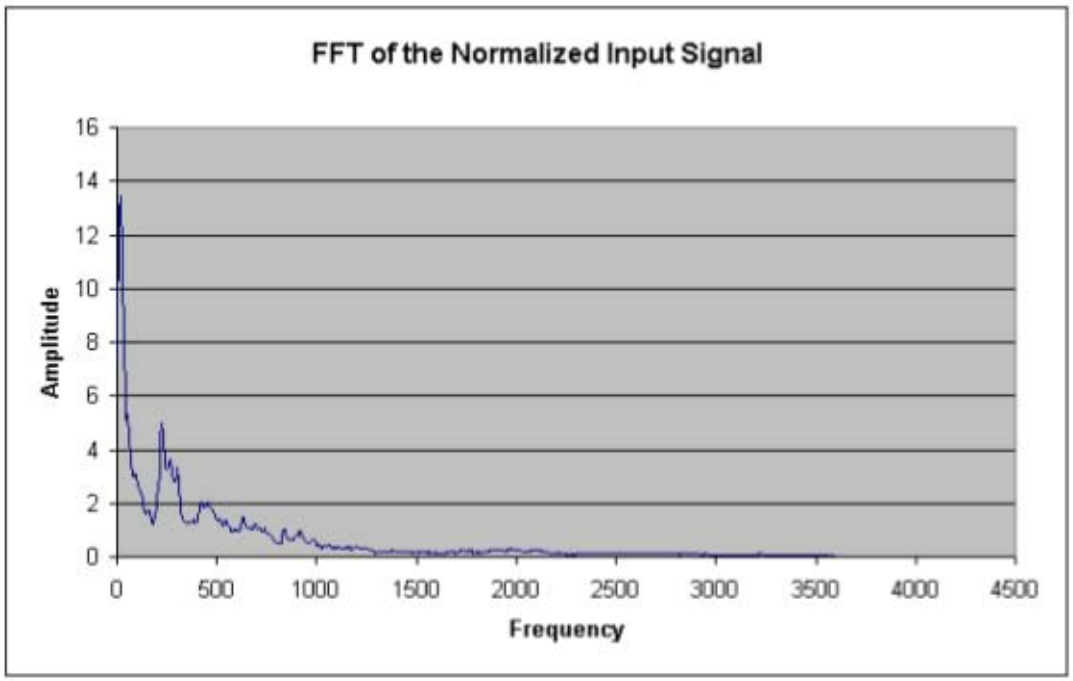

Figure 2.5: Fast Fourier Transform [1]

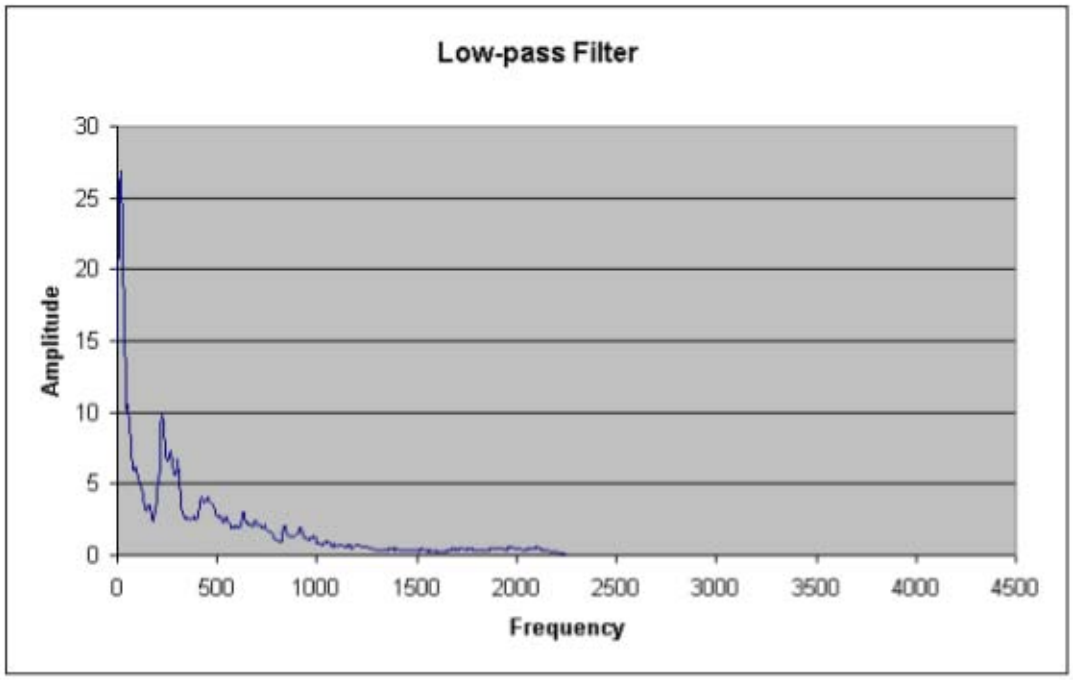

Figure 2.6: Low-Pass Filter [1]

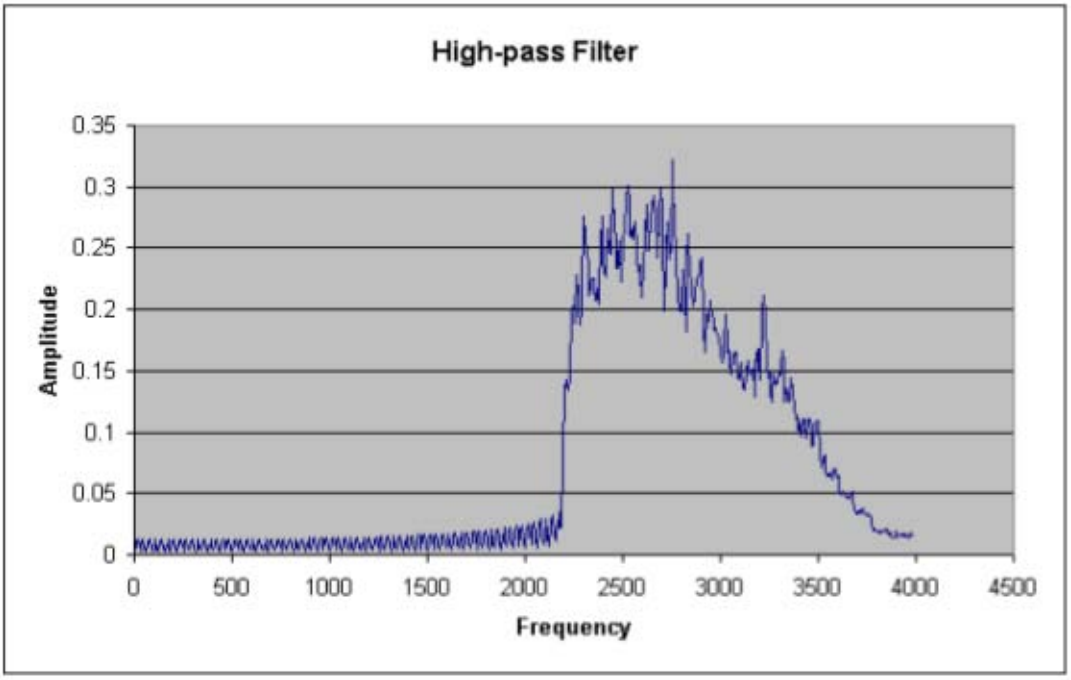

Figure 2.7: High-Pass Filter [1]

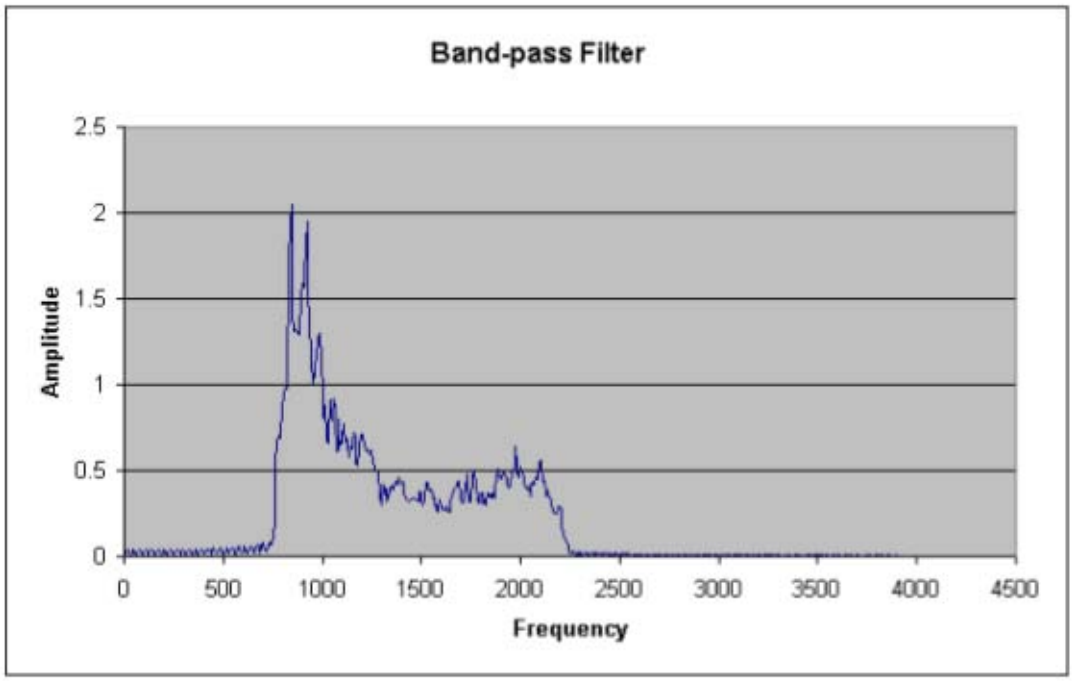

Figure 2.8: Band-Pass Filter [1]

CHAPTER 3:
Testing the Performance of the Modular Audio Recognition Framework

In this chapter, the performance of the Modular Audio Recognition Framework (MARF) in solving the open-set speaker recognition problem is described. MARF was tested for accuracy not speed. Accuracy was tested with variation along the following axes:

- Training set size
- Test sample size
- Background noise

First a description of the testing environment is given. It will cover the hardware and software used and discuss how they were configured so that the results can be replicated. Then the test results are described.

3.1 Test environment and configuration

3.1.1 Hardware

It is the beauty of this software solution that the only hardware required is a computer. The hardware used in experimentation was the author's laptop, a Dell Studio 15. The system is a 64-bit Mobile Intel 4 Series Express Chipset Family architecture fitted with the Intel T5800 CPU.

3.1.2 Software

The laptop is running the 64-bit version of the Arch Linux distribution (`http://www.archlinux.org/`). It is installed with a monolithic kernel, version 2.6.34. The sound card kernel module is snd_hda_intel. Advanced Linux Sound Architecture (ALSA) version 1.0.23 is used as the kernel level audio API. The current version of Sun Java install is the Java(TM) SE Runtime Environment (build 1.6.0_20-b02).

For the speaker recognition system software, the system contains the latest version of the Modular Audio Recognition Framework (MARF) version 0.3.0-devel-20100519-fat. It is installed as

a precompiled Java archive (jar) that exists in the system's CLASSPATH variable. The software that is responsible for the user recognition is the Speaker Identification Application (SpeakerIdentApp) which is packaged with MARF version 0.3.0-devel-20060226.

The SpeakerIdentApp can be run with with a preprocessing filter, a feature extraction setting, and a classification method. The options are as follows:

```
Preprocessing:
  -silence              - remove silence (can be combined with any below)
  -noise                - remove noise (can be combined with any below)
  -raw                  - no preprocessing
  -norm                 - use just normalization, no filtering
  -low                  - use low-pass FFT filter
  -high                 - use high-pass FFT filter
  -boost                - use high-frequency-boost FFT preprocessor
  -band                 - use band-pass FFT filter
  -endp                 - use endpointing

Feature Extraction:
  -lpc                  - use LPC
  -fft                  - use FFT
  -minmax               - use Min/Max Amplitudes
  -randfe               - use random feature extraction
  -aggr                 - use aggregated FFT+LPC feature extraction

Pattern Matching:
  -cheb                 - use Chebyshev Distance
  -eucl                 - use Euclidean Distance
  -mink                 - use Minkowski Distance
  -mah                  - use Mahalanobis Distance
```

There are 19 prepossessing filters, five types of feature extraction, and six pattern matching methods. That leaves us with $19 \times 5 \times 6 = 570$ permutations for testing. To facilitate this, we used a bash script that would run a first pass to learn all the speakers using all the above permutations then test against the learned database to identify the testing samples. The script can be found in Appendix section A. Please note the command-line options correspond to some

of the feature extraction and classification technologies discussed in Chapter 2.

Other software used: Mplayer version SVN-r31774-4.5.0 for conversion of the 16-bit PCM wav files from 16kHz sample rate to Mono, 8kHz, 16-bit sample which is what SpeakerIdentApp expects. Gnu SoX v14.3.1 was used to trim testing audio files to desired lengths.

3.1.3 Test subjects

In order to allow for repeatable experimentation, all "users" are part of the MIT Mobile Device Speaker Verification Corpus [19]. This is a collection of 21 female and 25 males voices. They are recorded in multiple environments. These environments are an office, a noisy indoor court ("Hallway"), and a busy traffic intersection. An advantage to this corpus is that not only is each user recorded in these different environments, but in each environment they utter one of nine unique phrases. This allows the tester to rule out possible erroneous results for a mash-ups of random phrases. Also, since these voices were actually recorded in their environments, not simulated, this corpus contains the Lombard effect, the fact speakers alter their style of speech in noisier conditions in an attempt to improve intelligibility[12].

This corpus also contains the advantage of being recorded on a mobile device. So, all the internal noise to the device can be found in the recording samples. In fact, Woo's paper contains a spectrograph showing this noise embedded in the audio stream [12].

The samples come as mono, 16-bit, 16kHz wav files. To be used in MARF, they must be converted to an 8kHz wav file. To accomplish this, Mplayer was run with the following command to convert the wav file to a MARF appropriate file using:

```
$ mplayer \
    -quiet
    -af volume=0,resample=8000:0:1 \
    -ao pcm:file="<fileForMARF>.wav" <initPCMfile>.wav
```

3.2 MARF performance evaluation

3.2.1 Establishing a common MARF configuration set

Before evaluating the performance of MARF along the three axes, it was necessary to settle on a common set of MARF configurations to be used in investigating performance across the three

axes. The configurations has three different facets of speaker recognition: 1) preprocessing, 2) feature extraction and 3) pattern matching or classification. Which configurations should be used? The MARF user's manual suggested some which have performed well. However, in the interest of testing the manual's hypotheses, we decided to see which configurations did the best with the MIT Corpus office samples and our testing machine platform.

We prepped all files in the MIT corpus file `Enroll_Session1.tar.gz` as outlined above. Then female speakers F00–F04 and male speakers M00-M04 were selected from the corpus as our training subjects. For each speaker, the "Office – Headset" environment was used. It was decided to initially use five training samples per speaker to initially train the system. The respective `phrase01` – `phrase05` was used as the training set for each speaker. The Speaker Identification Application was then run to both learn the speakers' voices and to test speaker samples. For testing, each speaker's respective `phrase06` and `phrase07` was used.

The output of the script given in A was redirected to a text file then manually put in an Excel spreadsheet to analyze. Using the MARF Handbook as a guide toward performance, we closely examined all results with the pre-prossessing filter `raw` and `norm` and with the pre-prossessing filter `endp` only with the feature extraction of `lpc`. With this analysis, the top-5 performing configurations were identified (see Table 3.1). For "Incorrect", MARF identfied a speaker other than the testing sample.

Table 3.1: "Baseline" Results

Configuration	Correct	Incorrect	Recog. Rate %
-raw -fft -mah	16	4	80
-raw -fft -eucl	16	4	80
-raw -aggr -mah	15	5	75
-raw -aggr -eucl	15	5	75
-raw -aggr -cheb	15	5	75

It is interesting to note that the most successful configuration of "-raw -fft -mah" was ranked as the 6th most accurate in the MARF user's manual from the testing they did runnung a similar script with their own speaker set[1]. These five configurations were then used in evaluating MARF across the three axes.

It should be pointed out that during identification of a common set of MARF configrations, it was discovered that MARF repeatedly failed to recognize a speaker for whom it was never

Table 3.2: Correct IDs per Number of Training Samples

	7	5	3	1
-raw -fft -mah	15	16	15	15
-raw -fft -eucl	15	16	15	15
-raw -aggr -mah	16	15	16	16
-raw -aggr -eucl	15	15	16	16
-raw -aggr -cheb	16	15	16	16

given a training set. From the MIT corpus, four "Office–Headset" speakers from the file `Imposter.tar.gz`, two male and two female(IM1, IM2, IF1, IF2), were tested against the set of *known* speakers. MARF failed to detect all four as unknown. Four more speakers were added in the same fashion above(IM3, IM4, IF3, IF4). Again, MARF failed to correctly identify them as an impostor. MARF consistanly issued false positives for all *unknown* speakers.

MARF is capible of outputting "*Unknown*" for user ID. For some configurations (that performed terribly) such as `-low -lpc -nn`, known speakers were displayed as *Unknown*. There is some threshold in place but whether it can be tuned is not documented. For this reason, further investigation of MARF along the three axes was limited to its performance in solving the closed-set speaker recognition problem.

3.2.2 Training-set size

As stated previously, the baseline was created with five training samples per user. We would like to see what is the minimum number of samples need to keep our above mentioned setting still accurate. We re-ran all testing with samples per user in the range of seven, five(baseline), three, and one. For each iteration, all MARF databases were flushed, feature extraction files deleted, and users retrained. Please see Table 3.2.

It is interesting to note that a set size of three actually produced the best results for MARF. Due to this discovery, the training set size of three will be the new baseline for the rest of testing.

3.2.3 Testing sample size

With a system as laid out in Chapter 4, it is critical to know how much voice data does MARF actually need to perform adequate feature extraction on the sample for voice recognition. We may need to get by with a shorter sample if, in real life the user talking gets cut off. Also, if the sample is quite long, it would allow us to break the sample up into many smaller parts

for dynamic re-testing, allowing us the ability to test the same voice sample multiple for higher accuracy. The voice samples in the MIT corpus range from 1.6 – 2.1 seconds in length. We have kept this sample size for our baseline, connoted as `full`. Using the gnu application SoX, we trimmed off the ends of the files to allow use to test the performance of our reference settings at the following lengths: `full`, `1000ms`, `750ms`, and `500ms`. Please see Graph 3.1 for the results.

SoX script as follows:

```
#!/bin/bash

for dir in `ls -d */*/`
do
  for i in `ls $dir*.wav`
  do
     newname=`echo $i|sed 's/.wav/_1000.wav/g'`
     sox $i $newname trim 0 1.0
     newname=`echo $i|sed 's/.wav/_750.wav/g'`
     sox $i $newname trim 0 0.75
     newname=`echo $i|sed 's/.wav/_500.wav/g'`
     sox $i $newname trim 0 0.5
  done
done
```

As shown in the graph, the results collapse as soon as we drop below 1000ms. This is not surprising, for as noted in Chapter 2, one really needs about 1023ms of data to perform ideal feature extraction.

3.2.4 Background noise

All of our previous testing has been done with samples made in noise-free environments. As stated earlier, the MIT corpus, includes recording made in noisy environments. For testing in this section, we have kept the relatively noise-free samples as our training-set and have included noisy samples to test against it. Recordings are taken from a hallway and an intersection. Graph 3.2 Show the effects of noise on each of our testing parameters.

What is most surprising is the severe impact noise had on our testing samples. More testing

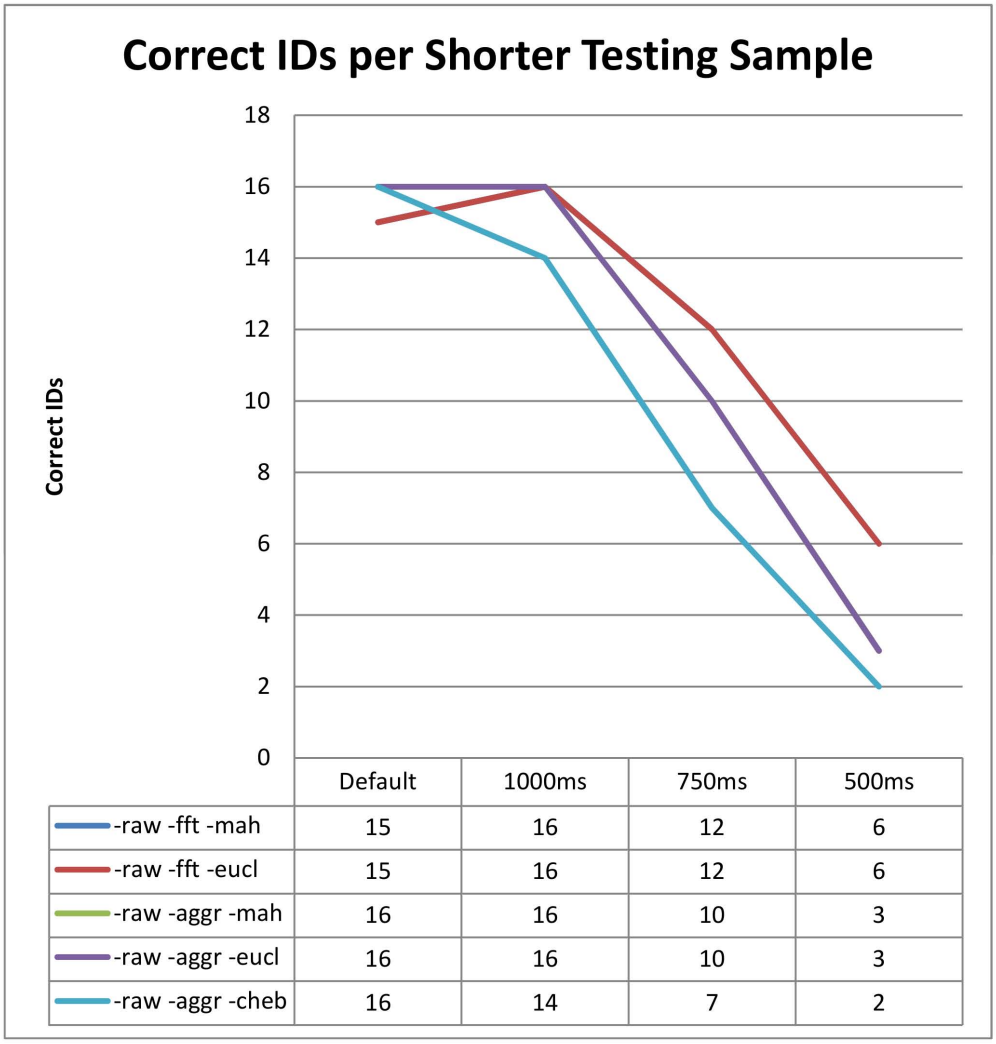

Figure 3.1: Top Setting's Performance with Variable Testing Sample Lengths

must to be done to see if combining noisy samples into our training-set allows for better results.

3.3 Summary of results

To recap, by using an available voice corpus, we were able to perform independently repeatable testing of the MARF platform for user recognition. Our corpus allowed us to account for both the Lombardi effect and the internal noise generated by a mobile device in our measurement. Starting with a baseline of five samples per user, we were able to extend testing to various parameters. We tested against adjustments to the user training-set to find the ideal number of training samples per user. From there we tested MARF's effectiveness at reduced testing sample length. Finally, we tested MARF's performance of samples from noisy environments.

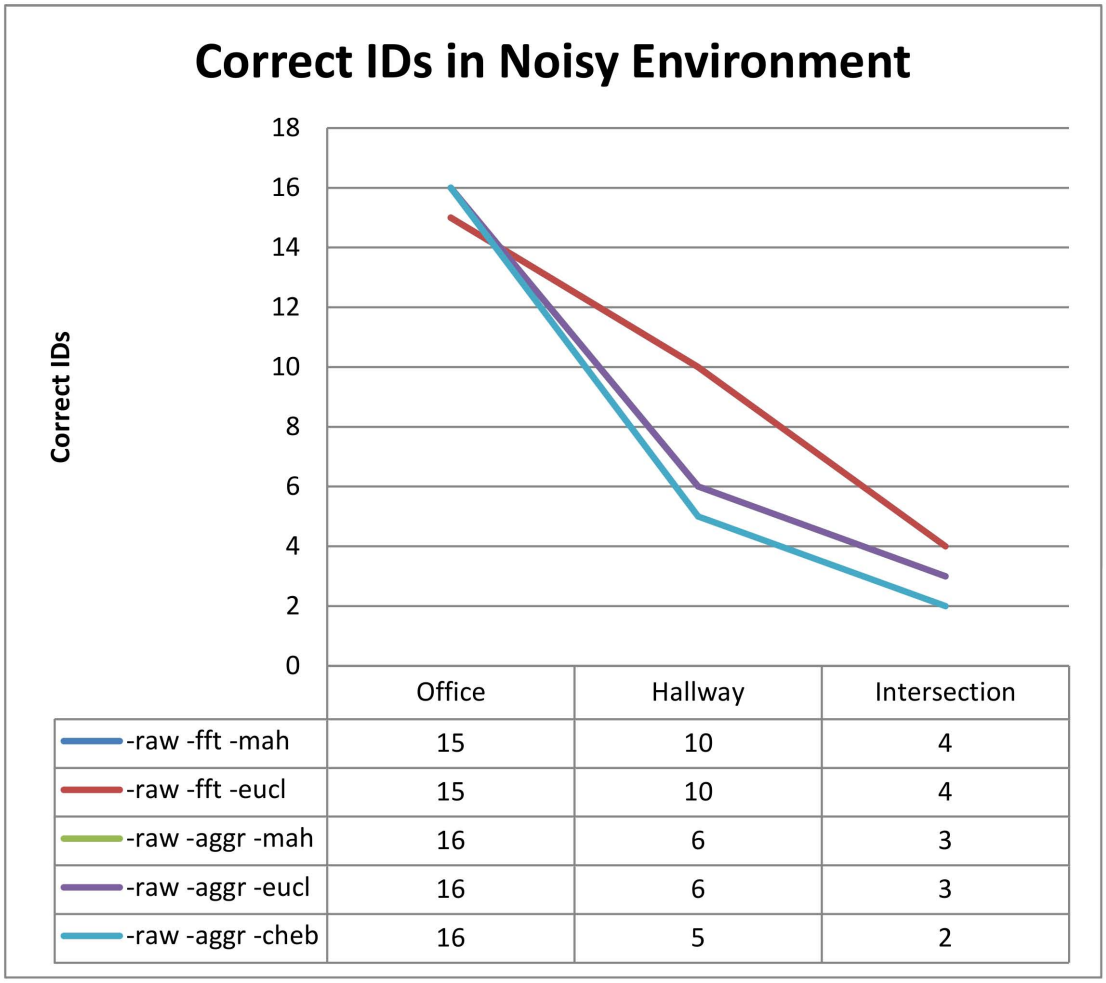

Figure 3.2: Top Setting's Performance with Environmental Noise

Testing proved that the Modular Audio Recognition Framework with its Speaker Identification Application succeeded at basic user recognition. MARF was also successful at recognizing users from sample lengths as short as 1000ms. This testing shows that MARF is a viable platform for speaker recognition.

The biggest failure with our testing was SpeakerIdentApp's inability to recognize an unknown user. In the top 20 testing results for accuracy, *Unknown User* was not even selected as the second guess. With this current shortcoming, it is not possible to deploy this system, as envisioned in Chapter 1, to the field. Since SpeakerIdentApp always maps a known user to a voice, we would be unable to detect a foreign presence on our network. Furthermore, it would confuse any type of Personal Name System we set up since the same user could get mapped to multiple phones as SpeakerIdentApp misidentifies an unknown user to a know user already bound to

another device. This is a huge shortcoming for our system.

MARF also performed poorly with a testing sample coming from a noisy environment. This is a critical shortcoming since most people authenticating with our system described in Chapter 4 will be contacting from a noisy environment, such as combat or a hurricane.

3.4 Future evaluation

3.4.1 *Unknown User* Problem

Due to the previously mentioned failure, more testing need to be done to see if SpeakerIdentApp can identify unknown voices and keep its 80% success rate on known voices. The MARF manual states better success with their tests when the pool of registered users was increased [1]. More tests should be done with a large group of speakers for the system to learn.

If more speakers do not increase SpeakerIdentApp's ability to identify unknown users, testing should also be done with some type of external probability network. This network would take the output from SpeakerIdentApp then try to make a "best guess" base on what SpeakerIdentApp is outputting and what it has previously outputted along with other information, such as, geo-location.

3.4.2 Increase Speaker Set

This testing was done with a speaker-set of ten speakers. More work needs to be done to explore the effects of increasing the number of users. For an accurate model of a real-world use of this system, SpeakerIdentApp should be tested with at least 50 trained users. It should be examined how the increased speaker set affects for trained user identification and unknown user identification.

3.4.3 Mobile Phone Codecs

While our testing did include the effect of the noisy EMF environment that is today's mobile phone, it lacked the effect caused by mobile phone codecs. This may be of significant consequence as work has shown the codecs used for GSM can significantly degrade the performance of speaker identification and verification systems [20]. Future work should include the effects of these codecs.

3.4.4 Noisy Environments

With MARF's failure with noisy testing samples, more work must be done to increase its performance under sonic duress. Wind, rain, and road noise along with other background noise most likely will severely impact SpeakerIdentApp's ability to identify speakers. As the creators of the corpus state, "Although more tedious for users, multistyle training (i.e. requiring a user to provide enrollment utterances in a variety of environments using a variety of microphones) can greatly improve robustness by creating diffuse models which cover a range of conditions[12]." This may not be practical for the environments in which this system is expected to operate.

CHAPTER 4:
An Application: Referentially-transparent Calling

This chapter sketches the design of a system in which the physical binding of users to cellphones via speaker recognition is leveraged to provide a useful service called *referential transparency*. The system is envisioned for use in a small user space, say less than 100 users, where every user must have the ability to call each other by name or pseudonym (no phone numbers). On the surface, this may not seem novel. After all, anyone can dial a friend by name today using a directory service that maps names to numbers. What is being proposed here is much different. Suppose a person makes some number of outgoing calls over a variety of cell phones during some period of time. At any time, this person may need to receive an incoming call, however, they have made no attempt to update callers of the number at which they can be currently reached. The system described here would put the call through to the cell phone at which the person made their most recent outbound call.

Contrast this process with that which is required when using a VOIP technology such as SIP. Certainly with SIP discovery, all users in an area could be found and phone books dynamically updated. But, what would happen if that device is destroyed or lost? The user needs to find a new device, deactivate whomever is logged into the device, then log themselves in. This is not at all passive, and in a combat environment, an unwanted distraction.

Finally, the major advantage of this system over SIP is the ability of many-to-one binding. It is possible with our system to have many users bound to one device. This would be needed if two or more people are sharing the same device. This is currently impossible with SIP.

Managing user-to-device bindings for callers is a service called referential transparency. This service has three major advantages:

- It uses a passive biometric approach, namely speaker recognition, to associate a person with a cell phone. Therefore callees are not burdened with having to update forwarding numbers.

- It allows GPS on cellular phones to be leveraged for determining location. GPS alone is inadequate since it indicates phone location and a phone may be lost or stolen.

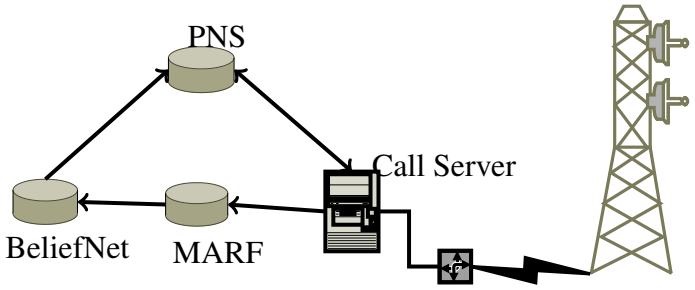

Figure 4.1: System Components

- It allows calling capability to be disabled by person rather than by phone. If an unauthorized person is using a phone then service to that device should be disabled until an authorized user uses it again. The authorized user should not be denied calling capability merely because an unauthorized user previously used it.

The service has many applications, including military missions and civilian disaster relief.

We begin with the design of the system and discuss its pros and cons. Lastly, we shall consider a peer-to-peer variant of the system and look at its advantages and disadvantages.

4.1 System Design

The system is comprised of four major components:

1. Call server - call setup and VOIP PBX.
2. Cellular base station - interface between cellphones and call server.
3. Caller ID - belief-based caller ID service.
4. Personal name server - maps a caller's ID to an extension.

The system is depicted in Figure 4.1.

Call Server

The first component we need is the call server. Each voice channel, or stream, must go through the call server. Each channel is half-duplex, that is, only one voice is on the channel. It is the call server's responsibility to mux the streams to and push them back out to the devices to create a conversation between users. It can mux any number of streams, from a one-to-one phone call to large group conference call. An example of a call server is Asterisk [21].

Cellular Base Station

The basic needs for a mobile phone network are the phones and some type of radio base station to which the phones can communicate. Since our design has off-loaded all identification to our caller-id system and is in no way dependent on the phone hardware, any mobile phone that is compatible with our radio base station can be used. This gives great flexibility in the procurement of mobile devices. We are not tied to any type of specialized device that must be ordered via the usual supply chains. Assuming we set up a GSM network, we could buy a local phone and tie it to our network.

With an open selection for devices, we have an open selection for radio base stations. The selection of a base station will be dictated solely by operational considerations as opposed to what technology into which we are locked. A commander may wish to ensure their base station is compatible with local phones to ensure local access to devices. It is just as likely, say in military applications, one may want a base station that is totally incompatible with the local phone network to prevent interference and possible local exploitation of the network. Base station selection could be based on what your soldiers or aid workers currently have in their possession. The decision on which phones or base stations to buy is solely dictated by operational needs.

Caller ID

The caller ID service, dubbed BeliefNet, is a probabilistic network capable of a high probability user identification. Its objective is to suggest the identity of a caller at a given extension. It may be implemented in general as a Bayesian network with inputs from a wide variety of attributes and sources. These include information such as how long it has been since a user was heard from on a device, the last device to which a user was associated, where they located the last time they were identified, etc. We could also consider other biometric sources as inputs. For instance, a 3-axis accelerometer embedded on the phone could provide a gait signature [22], or a forward-facing camera could provide a digital image of some portion of the person. The belief network operates continuously in the background, as it is supplied new inputs, constantly making determinations about caller IDs. It is invisible to callers. A belief network was not constructed as part of this thesis. The only attribute considered for this thesis was voice, specifically, its analysis by MARF.

As stated in Chapter 3, for MARF to function it needs both a training set (set of known users) and a testing set (set of users to be identified). The training set would be recorded before a team

member deployed. It could be recorded either at a PC, as done in Chapter 3, or it could be done over the mobile device itself. The efficacy of each approach will need to be tested in the future. The voice samples would be loaded onto the MARF server along with a flat-file with a user id attached to each file name. MARF would then run in training mode, *learn* the new users, and be ready to identify them at a later date.

The call server may be queried by MARF, either via Unix pipe or UDP message (depending on the architecture). The query requests a specific channel and a duration of time of sample. If the channel is in use, the call server returns to MARF the requested sample. MARF attempts to identify the voice on the sample. If MARF identifies the sample as a known user, this user information is then pushed back to the call server and bound as the user id for the channel.

Should a voice be declared as *unknown*, the call server stops sending voice and data traffic to the device associated with the unknown voice. The user of the device can continue to speak and, quite possibly, if it was a false negative, be reauthorized onto the network without ever knowing they had been disassociated from the network. At anytime, the voice and data will flow back to the device as soon as someone *known* starts speaking on the device.

Caller ID running the BeliefNet will also interface with the call server, but where we install and run it will be dictated by need. It may be co-located on the same machine as the call server or may be many miles away on a sever in a secured facility. It could also be connected to the call server via a Virtual Private Network (VPN) or public lines if security is not a top concern.

Personal Name Service

As mentioned in Chapter 1, we can incorporate a type of Personal Name Service (PNS) into our design. We can think of this closely resembling Domain Name Service (DNS) found on the Internet today. As a user is identified, their name could be bound to the channel they are using in a PNS hierarchy to allow a dial by name service.

Consider the civilian example of disaster response. We may gave a root domain of `.flood`. Within that that disaster area we could have an aid station with near a river. This could be addressed as `aidstation.river.flood`. As aid worker "Bob" uses the network, he is identified by MARF and his device is now *bound* to him. Anyone is working in the domain of `aidstation.river.flood` would just need to dial "Bob" to reach him. Someone at flood command could dial `bob.aidstation.river` to contact him. Similar to the other services, PNS could be located on the same server as MARF and the call server, or, be located

on a separate machine connect via an IP network.

4.2 Pros and Cons

The system is completely passive from the caller's perspective. Each caller and callee is bound to a device through normal use via processing done by the caller ID sub-component. This is entirely transparent to both parties. There is no need to key in any user or device credentials.

Since this system may operate in a fluid environment where users are entering and leaving an operational area, provisioning users must not be onerous. All voice training samples are stored on a central server. It is the only the server impacted by transient users. This allows central and simplified user management.

The system overall is intended to provide referential transparency through a belief-based caller ID mechanism. It allows us to call parties by name, however, the extensions at which these parties may be reached is only *suggested* by the PNS. We do not know whether these are correct extensions as they arise from doing audio analysis only. Cryptography and shared keys cannot be relied upon in any way because the system must operate on any type of cellphone without a client-side footprint of any kind, as discussed in the next section, we cannot assume we have access to the kernel space of the phone. It is therefore assumed that these extensions will actually be dialed or connected to so that a caller can attempt to speak to the party on the other end and confirm their identity through conversation. Without message authentication codes, there is a man-in-the-middle threat that could place an authorized user's voice behind an unauthorized extension. This makes the system unsuitable for transmitting secret data to cellphones since they are vulnerable to intercept.

4.3 Peer-to-Peer Design

It is easy to imagine our needs being met with a simple peer-to-peer model without any type of background server. Each handset, with some custom software, could identify a user, bind their name to itself, push out this binding to the ad-hoc network of other phones running similar software, and allow its user to fully participate on the network.

This design does have several advantages. First, it is a simple setup. There is no need for a network infrastructure with multiple services. Each device can be pre-loaded with the users it expects to encounter for identification. Second, as the number of network users grow, one needs just to add more phones to the network. There would not be a back-end server to upgrade or

network infrastructure to build-out to handle the increase in MARF traffic. Lastly, due to this lack of back-end services, the option is much cheaper to implement. So, with less complexity, clean scalability, and low cost, could this not be a better solution?

There are several drawbacks to the peer-to-peer model that are fatal. First, user and device management becomes problematic as we scale up the number of users. How does one know which training samples are stored on which phones? While it would be possible to store all our *known* users on a phone, phone storage is finite; as our number of users grow, we would quickly run out of storage on the phone. Even if storage is not an issue, there is still the problem of adding new users. Every phone would have to be recalled and updated with the new user.

Then there is issue of security. If one of these phones is compromised, the adversary now has access to the identification protocol, and worse, multiple identification packages of *known* users. It could be trivial for an attacker the modify this system and defeat its identification suite, thus giving an attacker spoofed access to the network, albeit limited.

Finally, if we want this system to be passive, we would need to install software that runs in the kernel space of the phone, since the software would need to have access to the microphone at all times. While this is certainly possible with the appropriate software development kit (SDK) it would mean for each type of phone, looking at both hardware and software, and developing a new voice sampling application with the appropriate SDK. This would tie the implementation to a specific hardware/software platform which seems undesirable as it limits our choices in the communications hardware we can use.

This chapter has explored one system where user-device binding can be used to provide referential transparency. How the system might be used in practice is explored in the next chapter.

CHAPTER 5:
Use Cases for Referentially-transparent Calling Service

A system for providing a referentially-transparent calling service was described in Chapter 4. In this chapter, two specific use cases for the service are examined, one military, the other civilian. How the system would be deployed in each case and whether improvements are needed to support them will be discussed.

5.1 Military Use Case

One of the driving use cases for the system has been in a military setting. The system's properties, as discussed in Chapter 4, were in fact developed with military applications in mind. Of interest here is deployment of the system at the Marine platoon level where the service would be used by roughly 100 users for combat operations as well as search and rescue.

Imagine a Marine platoon deployed to an area with little public infrastructure. They need to set up communications quickly to begin effective operations. First, they would install their radio base station within a fire-base or area that is secure. All servers associated with the base station would likewise be stored within a safe area. The call and personal name servers would be installed behind the base station. As Marines come to the base for operations, their voices would be recorded via a trusted handheld device or with a microphone and laptop. MARF, co-located with the Call server, would then train on these voice samples.

As Marines go on patrol and call each other over the radio network, their voices are constantly sampled by the Call server and analyzed by MARF. The Personal Name server is updated accordingly with a fresh binding that maps a user to a cell phone number. This process is ongoing and occurs in the background. Along with this update, other data may be stored on the Name server such a GPS data and current mission. This allows a commander, say the Platoon Leader at the fire-base, to monitor the locations of Marines on patrol, and to get a picture of their situation by monitoring overall communications on the Call server. Since the Platoon Leader would have access to the Call server, mission updates (e.g. a change in patrol routes, mission objective, etc.) could be managed there as well. With the Personal Name system, alerts could be made by simply calling `platoon1` or `squad1.platoon1` for example.

At some point, the members of a platoon may engage in battle which could lead to lost or damaged cell phones. Any phones that remain can be used by the Marines to automatically refresh their cell phone bindings on the Name server via MARF. If a squad leader is forced to use another cell phone then the Call server will update the Name server with the leader's new cell number automatically. Calls to the squad leader now get sent to the new number without ever having to know the new number.

Marines may also get separated from the rest of their squad for many reasons. They may even be wounded or incapacitated. The Call and Name servers can aid in the search and rescue. As a Marine calls in to be rescued, the Name server at the firebase has their GPS coordinates. Furthermore, MARF has identified the speaker as a known Marine. Both location and identity have been provided by the system. The Call server can even indicate from which Marines there has not been any communications recently, possibly signalling trouble. For instance, the platoon leader might be notified after a firefight that three Marines have not spoken in the past five minutes. That might prompt a call to them for more information on their status.

5.2 Civilian Use Case

The system was designed with the flexibility to be used in any environment where people need to communicate with each other. The system is flexible enough to support disaster response teams. An advantage to using this system in a civilian environment is that it could be stood up in tandem with existing civilian telecommunications infrastructure. This would allow for immediate operations in the event of a disaster as long as cellular towers are operating. Each civilian cell tower, or perhaps a geographic group of towers, could be serviced by a cluster of Call servers. Ideally there would also be redundancy, or meshing, of the towers so that if a Call server went down, there would be a backup for the orphaned cell towers.

Call servers might also be organized in a hierarchical fashion as was described in Chapter 1. For instance, there might be a Call server for the *North Fremont* area. Other servers placed in local areas could be part of a larger group, say *Monterey Bay*. This, with other regional servers could be grouped with *SF Bay* which would be part of *Northern California*, etc. This hierarchical structure would allow for a state disaster coordinator to direct-dial the head of an affected region. For example, one could dial `boss.nfremont.mbay.sfbay.nca`. Though work has been done to extend communications systems by way of portable, ad-hoc wide-area networks (WANs) [23] for civilian disaster response, the ability for state-level disaster coordinators to immediately reach people on the ground using the current civilian phone infrastructure is un-

precedented in U.S. disaster response.

For the purpose of disaster response, it may be necessary to house the Call servers in a hardened location with backup power. Unfortunately, cell towers are far more exposed and cannot be protected this way and hence they may become inoperable due to damage or loss of power. However, on the bright side, telcos have a vested interest in getting their systems up as soon as possible following a disaster. A case in point is the letter sent to the FCC from Cingular Communications following Hurricane Katrina in which the company acknowledges the importance of restoring cellular communications:

> The solutions are: generators to power the equipment until commercial power is restored, fuel to power the generators, coordination with local exchange carriers to restore the high speed telecommunications links to the cell sites, microwave equipment where the local wireline connections cannot be restored, portable cell sites to replace the few sites typically damaged during the storm, an army of technicians to deploy the above mentioned assets, and the logistical support to keep the technicians fed, housed, and keep the generators, fuel, and equipment coming.[24]

Katrina never caused a full loss of cellular service and within one week most of the service had been restored [24]. With dependence on the cellular providers to work in their interest to restore cell service, along with implementation of an *Emergency Use Only* cell-phone policy in the hardest hit areas, the referentially-transparent call system would be fairly robust.

MARF could be trained with disaster-response personnel via the Call server. As part of responder preparation, local disaster response personnel would already be *known* to the system. As the disaster becomes unmanageable for local responders, state government, and possibly national assets, would be called into the region. As they move in, their pre-recorded voice samples, stored on their respective servers, would be *pushed* to MARF via the Call server. In the worst case, these samples would be brought on a CD-ROM disc or flash drive to be manually loaded onto the Call server. As their samples are loaded onto the new servers, their IDs would contain their Fully Qualified Personal Name (FQPN). So when *Sally* is identified speaking on a device in the Seventh Ward of New Orleans, the FQPN of `sally.cell.tech.usace.us` gets bound to her current device as does `sally.seven.ward.no.la`.

The disaster-response use case relies heavily on integration with civilian communications systems. Currently no such integration exists. There are not only technical hurdles to overcome but

political ones as well. Currently the Department of Homeland Security is looking to build-out a national 700 MHz communications network [25]. Yet, James Arden Barnett, Jr., Chief of the Public Safety and Homeland Security Bureau, argues that emergency communications should link into the new 4G networks being built [26], showing that the FCC is really beginning to address federal communications integration with public infrastructure.

The use case also relies on the ability to shut off non-emergency use of the cell phone network. Though the ability to *shut off* non-emergency calling currently does not exist, calling priority systems are in place [27]. Currently, government officials who have been issued a Government Emergency Telecommunications Systems (GETS) card may get priority in using the public switched network (PSN)[28]. Similarly, the Wireless Priority Service (WPS) has also been setup by the National Communications Systems (NCS) agency. Both systems proved effective during Hurricane Katrina [29] and show that cell phone use for emergency responders is a reliable form of communication after a natural disaster.

CHAPTER 6:
Conclusion

This thesis has not only shown the viability of user recognition with voice as the biometric, but has shown how it can be effectively used for both combat and civilian applications. We have looked at the technology that comprises and the current research being done on speaker recognition. We have examined how this technology can be used in a software package, such as MARF, to have practical results with speaker recognition. We examined how speaker recognition with MARF could fit within a specific system to allow for passive user binding to devices. Finally in the previous chapter we examined what deployment of these systems would look like with regards to both military and civilian environments.

Speaker recognition is the most viable biometric for user-to-device binding due to its passivity and its ubiquitous support on all voice communications networks. This thesis has laid out a viable system, worthy of further research. Both Chapters 3 and 4 show the effectiveness of this system and that it is indeed possible to construct. Chapter 5 demonstrated that, in the abstract, this system can be used in both a military and civilian environment with a high expectation of success.

6.1 Road-map of Future Research

This thesis focused on using speaker recognition to passively bind users to their devices. This system is not only comprised of a speaker recognition element, but a Bayesian network dubbed BeliefNet. Discussion of the network comprised the use of other inputs for the BeliefNet, such as geolocation data.

Yet, as discussed in Chapter 4 no such BeliefNet has been constructed. There is a significant amount of research that needs to be done in this area to decide on the ideal weights of all our inputs and how their values effect each other. Successful research has been done at using such a Bayesian network for improving speech recognition with both audio and video inputs [30].

So, far we have only discussed MARF as the only input into our BeliefNet, but, what other data could we feed into it? We discussed in both Chapters 4 and 5, feeding in other data, such as the geo-location data from the cell phone. But there are many areas of research to enhance our system by way of the BeliefNet.

Captain Peter Young, USMC, has done work at the Naval Postgraduate School to test the effectiveness of detecting motion from the ground vibrations caused by walking using the accelerometers on the Apple iPhone [31]. Further work could be done to use this same technology to detect and measure human gait. As more research is done of how effective gait is as a biometric, we can imagine how the data from the accelerometers of the phone, along with, geo-location, and of course, voice, could all be fed into the BeliefNet to make its associations of users-to-device more accurate.

Along with accelerometers found in most smartphones, it is almost impossible to find a cell phone without a built in camera. The newest iPhone to market actually has a forward facing camera, that is, as one uses the device, they can have the camera focus on their face. Already work has been done focusing on the feasibility of face recognition on the iPhone [32]. So leveraging this work we have yet another information node on our BeliefNet.

As discussed in Chapter 3, the biggest shortcoming we currently have is that of MARF issuing false positives. Continued research must be done to allow to narrow MARF's thresholds for a *positive* identification.

As also discussed in Chapter 3, more work needs to be done on MARF's ability to process a large speaker databases, say on the order of several hundred. If the software cannot cope with such a large speaker group, is there possible ways the thread MARF to examine a smaller set? Would this type of system need to be distributed over multiple disks, computers?

6.2 Advances from Future Technology

Technology is constantly changing. This can most obviously be seen with the advances in smartphones over in that last three years. The original iPhone was a 32-bit RISC ARM running at 412MHz, supporting 128MB of RAM, and a two megapixel camera. One of the newest smartphones, the HTC Desire, comes with a 1 GHz Snapdragon processor, an AMD Z430 graphics processing unit (GPU), 576MB of RAM, and a five megapixel camera with autofocus, LED flash, face detection, and geotagging in picture metadata. No doubt, the Desire will be obsolete as of this reading. It is clear that as these devices advance, they could take the burden off the system described in Chapter 4 by allowing the phone to do more processing on-board with the phone's own organic systems. These advances in technology would not only change the design of the system, but could possibly positively affect performance.

There could also be advances in digital signal processing (DSP) that would allow the func-

tions of MARF to run directly in hardware. Already research has been done by the Wearable Computer Lab in Zurich Switzerland on using a DSP system that can be worn during daily activities for speaker recognition [33]. Given the above example of the technological advances of cell phones, it is not inconceivable that such a system of DSPs could exist within a future smartphone. Or, more likely, this DSP system could be co-located with the servers for our user-to-device binding system, alleviating the computational requirements for running MARF.

6.3 Other Applications

The voice recognition testing in this thesis could be used in other applications besides user-to-device binding. Since we have demonstrated the initial effectiveness of MARF in identifying speakers, it is possible to expand this technology to many types of telephony products.

We could imagine its use in a financial bank call center. One would just need to call the bank, have their voice sampled, then could be routed to a customer service agent who could verify the user. All this could be done without ever having the user input sensitive data such as account or social security numbers. This is an idea that has been around for sometime[34], but an application such as MARF may bring it to fruition.

THIS PAGE INTENTIONALLY LEFT BLANK

REFERENCES

[1] The MARF Reseach and Development Group. *Modular Audio Recognition Framework and its Applications*, 0.3.0.6 (0.3.0 final) edition, December 2007.

[2] M. Bishop. Mobile phone revolution. `http://www.developments.org.uk/articles/loose-talk-saves-lives-1/`, 2005. [Online; accessed 17-July-2010].

[3] D. Cox. Wireless personal communications: What is it? *IEEE Personal Communications*, pp. 20–35, 1995.

[4] Y. Cui, D. Lam, J. Widom, and D. Cox. Efficient pcs call setup protocols. Technical Report 1998-53, Stanford InfoLab, 1998.

[5] S. Li, editor. *Encyclopedia of Biometrics*. Springer, 2009.

[6] J. Daugman. Recognizing persons by their iris patterns. In *Biometrics: personal identification in networked society*, pp. 103–122. Springer, 1999.

[7] A.M. Ariyaeeinia, J. Fortuna, P. Sivakumaran, and A. Malegaonkar. Verification effectiveness in open-set speaker identification. *IEE Proc. - Vis. Image Signal Process.*, 153(5):618–624, October 2006.

[8] J. Pelecanos, J. Navrátil, and G. Ramaswamy. Conversational biometrics: A probabilistic view. In *Advances in Biometrics*, pp. 203–224. London: Springer, 2007.

[9] D.A. Reynolds. An overview of automatic speaker recognition technology. In *Acoustics, Speech, and Signal Processing, 2002. Proceedings.(ICASSP'02). IEEE International Conference on*, volume 4. IEEE, 2002. ISBN 0780374029. ISSN 1520-6149.

[10] D.A. Reynolds. Automatic speaker recognition: Current approaches and future trends. *Speaker Verification: From Research to Reality*, 2001.

[11] J.P. Campbell Jr. Speaker recognition: A tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, 2002. ISSN 0018-9219.

[12] R.H. Woo, A. Park, and T.J. Hazen. The MIT mobile device speaker verification corpus: Data collection and preliminary experiments. In *Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The*, pp. 1–6. IEEE, 2006.

[13] A.E. Cetin, T.C. Pearson, and A.H. Tewfik. Classification of closed-and open-shell pistachio nuts using voice-recognition technology. *Transactions of the ASAE*, 47(2):659–664, 2004.

[14] S.A. Mokhov. Introducing MARF: a modular audio recognition framework and its applications for scientific and software engineering research. *Advances in Computer and Information Sciences and Engineering*, pp. 473–478, 2008.

[15] J.F. Bonastre, F. Wils, and S. Meignier. ALIZE, a free toolkit for speaker recognition. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005), Philadelphia, USA*, pp. 737–740, 2005.

[16] K.F. Lee, H.W. Hon, and R. Reddy. An overview of the SPHINX speech recognition system. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(1):35–45, 2002. ISSN 0096-3518.

[17] S.M. Bernsee. The DFT" à Pied": Mastering The Fourier Transform in One Day, 1999, DSPdimension. com.

[18] J. Wouters and M.W. Macon. A perceptual evaluation of distance measures for concatenative speech synthesis. In *Fifth International Conference on Spoken Language Processing*, 1998.

[19] MIT Computer Science and Artificial Intelligence Laboratory. MIT Mobile Device Speaker Verification Corpus. website, 2004. `http://groups.csail.mit.edu/sls/mdsvc/index.cgi`.

[20] L. Besacier, S. Grassi, A. Dufaux, M. Ansorge, and F. Pellandini. GSM speech coding and speaker recognition. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2. IEEE, 2002. ISBN 0780362934.

[21] M. Spencer, M. Allison, and C. Rhodes. The asterisk handbook. *Asterisk Documentation Team*, 2003.

[22] M. Hynes, H. Wang, and L. Kilmartin. Off-the-shelf mobile handset environments for deploying accelerometer based gait and activity analysis algorithms. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pp. 5187–5190. IEEE, 2009. ISSN 1557-170X.

[23] A. Meissner, T. Luckenbach, T. Risse, T. Kirste, and H. Kirchner. Design challenges for an integrated disaster management communication and information system. In *The First IEEE Workshop on Disaster Recovery Networks (DIREN 2002)*, volume 24. Citeseer, 2002.

[24] L. Fowlkes. Katrina panel statement, Febuary 2006.

[25] A. Pearce. An Analysis of the Public Safety & Homeland Security Benefits of an Interoperable Nationwide Emergency Communications Network at 700 MHz Built by a Public-Private Partnership. *Media Law and Policy*, 2006.

[26] Jr. J.A. Barnett. National Association of Counties Annual Conference 2010. Technical report, Federal Communications Commission, July 2010.

[27] B. Lane. Tech Topic 18: Priority Telecommunications Services. 2008. `http://www.fcc.gov/pshs/techtopics/techtopics18.html`.

[28] U.S. Department of Health & Human Services. HHS IRM Policy for Government Emergency Telecommunication System Cards Ordering, Usage and Termination, November 2002. `http://www.hhs.gov/ocio/policy/2002-0001.html`.

[29] P. McGregor, R. Craighill, and V. Mosley. Government Emergency Telecommunications Service(GETS) and Wireless Priority Service(WPS) Performance during Katrina. In *Proceedings of the Fourth IASTED International Conference on Communications, Internet, and Information Technology*. Acta Press Inc,# 80, 4500-16 Avenue N. W, Calgary, AB, T 3 B 0 M 6, Canada,, 2006. ISBN 0889866139.

[30] T. Yoshida, K. Nakadai, and H.G. Okuno. Automatic speech recognition improved by two-layered audio-visual integration for robot audition. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pp. 604–609. Citeseer, 2010.

[31] P.J. Young. A Mobile Phone-Based Sensor Grid for Distributed Team Operations. Master's thesis, Naval Postgraduate School, 2010.

[32] K. Choi, K.A. Toh, and H. Byun. Realtime training on mobile devices for face recognition applications. *Pattern Recognition*, 2010. ISSN 0031-3203.

[33] M. Rossi, O. Amft, M. Kusserow, and G. Troster. Collaborative real-time speaker identification for wearable systems. In *Pervasive Computing and Communications (PerCom), 2010 IEEE International Conference on*, pp. 180–189. IEEE, 2010.

[34] D. O'Shaughnessy. Speaker Recognition. *IEEE ASSP Magazine*, 1986.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A:
Testing Script

```
#!/bin/bash

#
# Batch Processing of Training/Testing Samples
# NOTE: Make take quite some time to execute
#
# Copyright (C) 2002 - 2006 The MARF Research and Development Group
#
# Converted from tcsh to bash by Mark Bergem
#
# $Header: /cvsroot/marf/apps/SpeakerIdentApp/testing.sh,v 1.37 2006/01/15
    20:51:53 mokhov Exp $
#

#
# Set environment variables, if needed
#

export CLASSPATH=$CLASSPATH:.:/usr/lib/marf/marf.jar
export EXTDIRS

#
# Set flags to use in the batch execution
#

java="java -ea -Xmx512m"
#set debug = "-debug"
debug=""
graph=""
#graph="-graph"
#spectrogram="-spectrogram"
spectrogram=""

if [ $1 == "--reset" ];
        then
        echo "Resetting Stats..."
```

```
        $java SpeakerIdentApp --reset
        exit 0
fi

if [ $1 == "--retrain" ];
    then
        echo "Training..."

        # Always reset stats before retraining the whole thing
        $java SpeakerIdentApp --reset

        for prep in -norm -boost -low -high -band -highpassboost -raw -endp
        do
                for feat in -fft -lpc -randfe -minmax -aggr
                do

                        # Here we specify which classification modules to
                            use for
                        # training. Since Neural Net wasn't working the
                            default
                        # distance training was performed; now we need to
                            distinguish them
                        # here. NOTE: for distance classifiers it's not
                            important
                        # which exactly it is, because the one of generic
                            Distance is used.
                        # Exception for this rule is Mahalanobis Distance,
                            which needs
                        # to learn its Covariance Matrix.

                        for class in -cheb -mah -randcl -nn
                        do
                                echo "Config: $prep $feat $class
                                    $spectrogram $graph $debug"
                                date

                                # XXX: We cannot cope gracefully right now
                                    with these combinations --- too many
                                # links in the fully-connected NNet, so run
                                     out of memory quite often; hence,
                                # skip it for now.
```

```
                    if [ "$class" == "-nn" ]; then
                        if [ "$feat" == "-fft" ] || [ "$feat" ==
                            "-randfe" ] || [ "$feat" == "-aggr"
                          ];
                    then
                        echo "skipping..."
                        continue
                        fi
                    fi

                    time $java SpeakerIdentApp --train training
                        -samples $prep $feat $class $spectrogram
                         $graph $debug
                done
            done
        done
fi

echo "Testing..."

for prep in -norm -boost -low -high -band -highpassboost -raw -endp
do
        for feat in -fft -lpc -randfe -minmax -aggr
        do
                for class in -eucl -cheb -mink -mah -diff -randcl -nn
                do
                        echo "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
                            "
                        echo "Config: $prep $feat $class $spectrogram
                            $graph $debug"
                        date
                        echo "============================================
                            "

                        # XXX: We cannot cope gracefully right now with
                            these combinations --- too many
                        # links in the fully-connected NNet, so run of
                            memeory quite often, hence
                        # skip it for now.
                        if [ "$class" == "-nn" ]; then
                            if [ "$feat" == "-fft" ] || [ "$feat" == "-
```

```
                randfe" ] || [ "$feat" == "-aggr" ]; then
                    echo "skipping..."
                    continue
                fi
            fi

            time $java SpeakerIdentApp --batch-ident testing-
                samples $prep $feat $class $spectrogram $graph
                $debug

            echo "------------------------------------------------
                "
        done
    done
done

echo "Stats:"

$java SpeakerIdentApp --stats > stats.txt
$java SpeakerIdentApp --best-score > best-score.tex
date > stats-date.tex

echo "Testing Done"

exit 0

# EOF
```

Referenced Authors

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California
3. Marine Corps Representative
   Naval Postgraduate School
   Monterey, California
4. Directory, Training and Education, MCCDC, Code C46
   Quantico, Virginia
5. Marine Corps Tactical System Support Activity (Attn: Operations Officer)
   Camp Pendleton, California